Как разделяют пингвинов

Сергей А. ЯРЕМЧУК, grinder@ua.fm

Как и любая другая операционная система, Linux обычно размещается на жестком диске компьютера, и пользователю, начиная с установки и заканчивая повседневным пользованием, приходится как-то взаимодействовать с разделами винчестера. Пришла пора разобраться, что все-таки творится там, на пингвиньих островах, с помощью чего можно их создавать, как они обозначаются, какие программы можно использовать для обслуживания файловых систем и, наконец, осмотреться во всем многообразии этих систем.

В статье я попробую ответить на все вопросы, которые когда-либо поступали на мой почтовый адрес по данной теме, а также на часто задаваемые в различных форумах. В некоторых случаях придется повторяться, исключительно для целостности восприятия общей картины и, естественно, для недавно присоединившихся. Также естественным желанием у меня было собрать всю информацию воедино. Статья будет логически разделена на четыре части. В первой рассмотрим все, что связано с разделами, от их наименования до количества. Во второй и третьей посмотрим, какие файловые системы можно использовать на созданных разделах. И в четвертой поговорим об утилитах, не рассмотренных в первых трех частях, и об оптимизации работы дисков.

Разделяй и властвуй

Начнем, пожалуй, с обозначения дисков, принятом в Linux. Традиционно в этой ОС ATA-диск (я думаю, SCSI уже как-то не актуален для десктопа) обозначается в соответствии с тем, к какому из интерфейсов он подключен: диск PrimaryIDE, подключенный как Master, всегда обозначается как /dev/hda, как Slave – hdb, соответственно, диск SecondaryIDE, подключенный как Master – hdd, и как Slave – hdc. Причем называться он так будет независимо от того, есть ли диск в устройстве в наличии на данный момент или нет. Так обозначается весь диск целиком. Но как и повелось в любой операционной системе, диск делится для удобства работы на разделы. Жесткий диск может иметь не более четырех первичных (Primary) разделов, которые в Linux всегда обозначаются цифрами от 1 до 4, например hda2 для второго первичного раздела первого IDE-мастера. Но кому-то одних только первичных разделов может показаться мало, поэтому нередко создают в одном из первичных так называемый расширенный (Extended) раздел, на котором в свою очередь создается несколько логических разделов, обозначаемых цифрами начиная с 5. При этом в Linux разделы можно создавать, как это принято в DOS/Windows, то есть расширенный раздел может быть создан только в одном из первичных! Например, на диске может быть три первичных раздела hda1-hda3 и несколько логических, начиная от hda5, которые размещаются на четвертом первичном. Напомню, что в BSD-системах логические разделы (BSD Partitions) можно создать внутри каждого первичного.

До недавнего времени описанная здесь система обозначения дисков считалась стопроцентно правильной и сомнению не подвергалась. Но появившиеся в последнее время дистрибутивы новой волны, такие как Gentoo, Lunar Linux внесли в свои коррективы в обозначение дисков. И виноваты в этом не сами их создатели, которые хотят запутать пользователя или как-то особенно выделиться. Нет, все обстоит иначе. Дело в том, что в ядрах Linux серии2.4.* появилась новая файловая система устройств devfs, которая избавляет от множества неприятностей и неудобств. В двух словах, так как это соответствует теме статьи, назначение ее таково. Как известно, в Linux все, что ни попадя, в том числе и различные устройства, являются файлами – это упрощение намного облегчает взаимодействие пользователя с системой, так как для работы с диском могут применяться те же команды, что и для работы с обычным текстовым файлом (.cat, .dd) причем это взаимодействие абсолютно одинаково для всех типов и марок дисков. Но все было бы хорошо, если бы не один момент. Для того чтобы ядро могло нормально распознавать устройства, специальные файлы устройств нумеруются двумя целыми числами – старшим (major number) и младшим (minor number). Первое из них соответствует типу устройства(например, 3 – это первый IDE-диск), а второе – конкретному устройству (0A – его десятый раздел)(подробнее о наименовании устройств посмотрите в

/usr/src/linux/Documentation/devices.txt). Так вот, major-номер не может быть присвоен от балды: если производитель хочет предложить свой драйвер для широкого использования, он должен входить в контакт с производителем ядер и получить для своего устройства «официальный» major-номер. И только после этого он может использоваться публично. Проблемы здесь две. Одна состоит в том, что каталог /dev буквально завален различными файлами устройств для возможной совместимости системы со всеми девайсами; по большому счету, если не предвидится серьезного апгрейда, лишние, конечно, можно и убрать, но не лучше ли их вообще туда не класть? Вторая состоит в следующем. На первоначальном этапе эта схема распределения номеров еще была, скажем так, вполне оправдана малым количеством доступных девайсов (на заре Интернета при распределении IP-номеров и имен службы DNS названия узлов и соответствующий адрес тоже просто заносились в файл hosts на каждом компьютере). Но при современных темпах почкования новых устройств все это вызывает головную боль как производителя, так и лиц, отвечающих за поддержку ядра. Да и Linux уже подошел к тому рубежу, когда все major-номера скоро будут исчерпаны. Выходом из всех этих проблем является использование devfs. При этом файлы устройств создаются «на лету» по мере подключения (хотя бывает, что для этого приходится систему все-таки перезагружать), при этом не захламляется каталог ненужными файлами, вдобавок, теперь, зайдя в /dev, можно узнать, какие устройства у вас присутствуют реально. После инициализации к устройствам обращаются по именам, поэтому необходимость в номерах отпала, со всеми вытекающими отсюда преимуществами. Хотя для обратной совместимости можно (но необязательно) указать major- и minor-номера. Так вот, в devfs по умолчанию используется совершенно другая номенклатура и предусмотрены иные каталоги для размещения файлов устройств. Так, в некоторых дистрибутивах файловая система устройств вообще монтируется в каталог /devices, а каталог /dev сохраняется для совместимости. Наши IDE-диски теперь можно найти в каталоге /dev/ide (SCSI – /dev/scsi), встроенному контролеру соответствует каталог /dev/ide/host0 (в платах с дополнительным контроллером доводилось видеть и каталог host1). Двум IDE-каналам этого контроллера соответствуют файлы /dev/ide/host0/bus0 и bus1, а подключенным дискам - каталоги /dev/ide/host0/bus0/target0 и target1. В каждом из этих каталогов имеется еще один lun0, в котором собственно и находятся файлы устройств, соответствующие как всему накопителю - disc, так и первичным (part1–part4) и логическим (part5–partN) его разделам. Исходя из этого, полное название дискового раздела будет выглядеть так: /dev/ide/host0/bus0/target0/lun0/part2. Поверьте мне, что когда я не смог найти привычного hda, то просто обалдел. Такое обозначение можно назвать логичным, понятным, но уж никак не удобным. Поэтому в некоторых дистрибутивах (в частности, Lunar Linux, Gentoo) используется более удобный вариант, предполагающий создание жестких ссылок – например, в файле /etc/fstab для обозначения приведенного в предыдущем примере раздела встречается уже совсем другое обозначение: /dev/discs/disc0/part2. Но для совместимости никто не запрещает создать символическую ссылку со старым обозначением и работать как ни в чем не бывало – в некоторых дистрибутивах это предусмотрено автоматически.

Итак, с обозначением разобрались. Следующий вопрос, постоянно мучающий читателя: сколько и какие разделы нужно создавать. Итак, внимание – для нормальной работы необходимо создать как минимум два раздела: первый системный – Linux native, второй раздел подкачки – Linux Swap. Под системный раздел желательно выделить, если вы предполагаете работать с X-Window, как минимум 800–1000Мб, но это, как вы понимаете, во многом зависит от самого дистрибутива: есть однодисковые, а есть 3-, 5- и даже 9-дисковые, так что разбирайтесь сами. Раздел подкачки желательно расположить, для увеличения скорости обмена данными, как можно ближе к началу диска, а идеальный вариант – на другом физическом диске. А лучше вообще разделить поровну между ними, сделав запись в /etc/fstab о равенстве их приоритетов:

```
/dev/hda5 swap swap defaults,pri=1 0 0
/dev/hdc5 swap swap defaults,pri=1 0 0
```

Но здесь, в зависимости от старости дистрибутива (ядра), могут быть свои ограничения на размер. В очень ранних дистрибутивах максимальный размер раздела подкачки не должен был превышать 16Мб, а максимальное количество таких разделов достигало восьми. В более поздних предел составлял уже 128Мб. Современные же ядра, начиная с 2.4.10, не могут монтировать swap,

если размер дискового раздела меньше 128Мб. Когда я это первый раз прочитал и посмотрел на свой 64-Мб swap, то не понял, в чем тут прикол. Вроде и так работает. Но ведь это официальная информация, а реально swap ограничен половиной адресного пространства оперативной памяти. Для i86-процессоров при размере страницы памяти 4Кб (значение по умолчанию) размер адресного пространства равен 4Гб, а максимальный размер swap, соответственно, – 2Гб. Такой вариант разбиения на два раздела, по-моему, – наиболее удобный вариант для новичка, во всяком случае мороки и проблем на этапе освоения будет поменьше. Затем, если пингвин приживется на вашем компьютере (только не стирайте сразу – у самого месяца три ушло на то, чтобы разобраться что к чему), желательно на отдельные разделы вынести каталог /home, в котором хранятся все пользовательские данные и настройки, а также раздел /usr/local в который по умолчанию устанавливаются все пользовательские программы, не входящие в дистрибутив штатно. В таком случае можно будет переустановить заново дистрибутив, не затрагивая при этом всех пользовательских настроек и не переустанавливая заново кучу программ, и пользоваться ими сразу после запуска (сравните с Windows). А что делать, если первоначально не были созданы все эти разделы, а теперь, по мере прочтения статьи, у вас созревает желание перенести их на отдельные разделы диска? Все очень просто до безобразия, создаем еще один раздел, затем монтируем в произвольную точку и просто копируем в него данные, а затем удаляем их из исходной папки, чтобы место не занимали. Например:

```
mount -t ext3 /dev/hdb3 /mnt/temp
# монтируем созданную файловую систему.
cp -R /usr/local/ /mnt/temp
# рекурсивно копируем в нее все из каталога /usr/local (это можно проделать и с
# помощью mc или другим удобным способом); здесь необходимо проследить за выводимыми
# сообщениями
umount /dev/hdb3
# размонтируем
mount -t ext3 /dev/hdb3 /usr/local
# монтируем поверх старого /usr/local (новые ядра позволяют это) для проверки, теперь
# прогоняем любимые программы
umount /dev/hdb3
rm -f /usr/local/*
# очищаем диск, так как информация все равно будет присутствовать
mount -t ext3 /dev/hdb3 /usr/local
# монтируем новый раздел в качестве /usr/local и наслаждаемся полученным результатом.
```

И теперь, чтобы новый раздел при загрузке каждый раз монтировался на свое место, прописываем в файл /etc/fstab следующие строки:

```
/dev/hdb3 /usr/local ext3 defaults 0 0
```

Вот за что я и люблю Linux. Все просто, никаких тебе реестров, где регистрируется каждая программа и потому при смене раздела или добавлении нового диска как правило приходится заново ее переустанавливать. Кстати, если есть сеть с постоянно работающими компьютерами, то можно данный раздел вообще расположить только на ОДНОМ из них, тем самым экономя место, резервируя возможность контролировать и централизованно наполнять его содержание, а монтировать его уже на этапе загрузки с помощью NFS; тогда и бэкапить легче будет. При этом пользователи даже не будут догадываться, что какой-то раздел находится где-то совсем на другом компьютере. К слову, в соответствии со стандартами, продвигаемыми Linux Standard Base (http://www.linuxbase.org), на смену каталогу /usr/local постепенно продвигается другой – /opt. И если первые два года моего знакомства с этой системой там было совсем пусто, то в последнее время он начал быстро и методично заполняться. Чтобы не создавать и здесь новый раздел, есть два выхода: явно задавать с помощью -prefix=/usr/local путь при компиляции таких программ, или, что удобнее всего, вместо каталога /opt просто создать символическую ссылку на /usr/local:

```
# ln -s /usr/local /opt
```

После этого все программы при установке будут попадать на созданный раздел диска.

Также не помешает создать еще один раздел – /boot, это самая критическая часть для загрузки операционной системы. И как я уже говорил в статье о загрузчиках, ядро не всегда можно загрузить с современных журналируемых файловых систем. После публикации статьи мне пришло много писем о том, что, мол, грузимся, и ничего. Но это ведь придумал не я, моя задача –

просто предупредить, что такой вариант вполне возможен, особенно на первых ядрах серии2.4. Кроме того, есть второе ограничение для старых дистрибутивов (в основном для тех, у кого ядро древнее 2.4 – пользователей Mandrake7, RedHat6 ит.д.), связанное с так называемой проблемой 1023-го цилиндра. Дело в том, что из-за ограничений, накладываемых BIOS'ом большинства Intel-совместимых компьютеров (заметьте, это ограничение именно BIOS'а, а не Linux'а), ядро может загружаться только с первых 1023цилиндров жесткого диска, причем если у вас их два, то возможно даже, только с первого мастер-диска (для справки: в Partition Magic есть даже ярлычок, указывающий на 1023-й цилиндр). Вся необходимая информация для загрузки LILO/GRUB, в том числе и ядро, находится в каталоге /boot. Поэтому чтобы выйти из сложившейся ситуации, расположите этот каталог в первых 1023цилиндрах (вариант загрузки с дискеты я откидываю сразу), а корневой / (и все остальные) раскидайте по диску (или дискам) так, как вам хочется. Монтировать данный раздел следует только при инсталляции нового ядра или новой схемы загрузки.

Запись в /etc/fstab может быть примерно следующей:

```
/dev/hda1 /boot ext2 noauto 1 2
```

Больше чем 20Мб данный раздел все равно не потянет, так что можно и раскошелиться. На серверах обычно дополнительно выносят в отдельный раздел каталоги /var и /tmp, но на домашнем компьютере вряд ли есть в этом какая-то необходимость. Из всего вышеизложенного вы должны понять, что создание разделов – процесс сугубо творческий, и ограничивает вас в этом только ваша собственная фантазия (и размеры диска(ов)).



Рис. 1

Итак, с обозначениями мы разобрались, теперь давайте посмотрим, с помощью чего все это можно сотворить.

При установке каждого дистрибутива Linux пользователь сталкивается с какой-нибудь более или менее удобной программой, с помощью которой можно разбить диск и

отформатировать его под нужную файловую систему. Особенно просты в применении графические утилиты **Disk Druid** в RedHat и **Hard Drake** в Mandrake (Рис.1).

Для тех, кто сталкивался хоть раз с Partition Magic и немного понимает суть процесса, особых проблем при их использовании возникнуть не должно. Но, господа (панове, товарищи), раздел под новую операционную сист ему должен быть предварительно подготовлен, т.е. очищен от информации, хотя это можно попробовать сделать и при установке, зайдя в другую консоль, и скопировать файлы в другой раздел. Не говорю уже о таком обычае как предварительное резервирование данных при всякой рискованной операции, к чему относится, конечно, и установка ЛЮБОЙ операционной системы. При установке же других дистрибутивов и в последующей деятельности придется общаться с совсем другими утилитами. Познакомимся с ними поближе.

До недавнего времени для работы с дисковыми разделами в уже установленном Linux я пользовался только одной программой, к которой привык с еще с долинуксовских времен - имя ей fdisk. Да, это та страшилка, которой пугают начинающего линуксоида на каждом сайте и книге. Она полностью удовлетворяла моим довольно скромным потребностям; к тому же и программы, аналогичной PM, в Linux все равно нет. Но естественно, есть и другие предназначенные для этого инструменты: cfdisk(кстати, применяется при установке Debian) и совсем новая GNU/parted, с которой я впервые столкнулся при установке Lunar Linux. В моем RedHat их не было, и для того чтобы не лазить за ними в Интернет, я их просто скопировал из раздела с Lunar Linux в аналогичный каталог RedHat; если программа требовала какую-то библиотеку, то поступал аналогично. В более тяжелых случаях (например, перенос KDE) я пользуюсь jail (http://www.gsyc.inf.uc3m.es/~assman/). Эта программка, правда, предназначена совсем для другой цели – создания chroot-среды на серверах и переноса в нее только нужных для работы программ, чтобы злоумышленник, взломав аккаунт, думал что он находится в настоящем корневом каталоге. Уж простите за маленькое отступление.

Итак, fdisk. Тех, кто пользовался хоть раз Windows-аналогом, найдут данную программу очень похожей. Позволю все-таки немного остановиться на ее работе, так как есть небольшие отличия. Запускается она с аргументом – названим диска, с которым будет производиться дальнейшая работа.

```
[root@grinder sergej]# /sbin/fdisk /dev/hdb
Command (m for help): m
```

Взаимодействие с пользователем осуществляется путем нажатия клавиш соответствующей требуемой операции. Как видно из примера, m (manual) выводит полную справку, **p** (print) – состояние разделов, **n** (new) создает раздел, **d** (delete) удаляет его (кстати, удалить раздел Linux можно только с помощью описываемых программ и PM, ДОСовский fdisk в данном случае абсолютно бесполезен). Все введенные команды будут выполнены только после ввода **w** (write), а при помощи **q** (quit) можно выйти без записи изменений.

```
Command (m for help): p
Disk /dev/hda: 255 heads, 63 sectors, 3649 cylinders
Units = cylinders of 16065 * 512 bytes

Device Boot Start  End Blocks  Id System
/dev/hda1   247  3649 27334597+ f  Win95 Ext'd (LBA)
/dev/hda2 *   1  246 1975963+  b Win95 FAT32
/dev/hda5   247  756 4096543+ 83 Linux
/dev/hda6   757  772 128488+  82 Linux swap
```

Далее, клавиша **a** включает флаг указания загрузочного раздела, с помощью **t** устанавливается типфайловой системы раздела - список всех доступных можно просмотреть с помощью опции **-l** (list) (Рис.2). Как видите, в списке присутствуют практически все широко известные файловые системы, но это не значит, что после выбора a5FreeBSD на данном участке диска тут же будет создана файловая система FFS. Нет, в данном случае разделу присваивается только идентификатор раздела, с помощью которого можно разве что зарезервировать тип, а сама файловая система будет создана только после его форматирования. Вспомните, в ДОС после fdisk всегда нужно было запускать format. Здесь ситуация аналогична, только программы для создания

файловой системы (форматирования) немного другие (и не одна). В нашем случае нас интересуют типы раздела Linux native(идентификатор 83), Linux swap(82), еще может пригодится тип5для создания расширенного раздела.

Итак, для создания раздела вводим n:

```
Command (m for help): n
Command action
l logical (5 or over)
p primary partition (1-4)
```

На первом этапе нужно определиться, какой раздел нужен, если первичный – жмем на **p**, расширенный – **l** (хотя в старых версиях встречалась и **e**). После ввода, когда запросят номер раздела, порядка придерживаться не обязательно, можно вначале создать второй, а потом первый. Следующий вопрос – о размере. Его можно задать в цилиндрах (по умолчанию), в байтах, килобайтах и мегабайтах. Если просто ввести цифру 100, то раздел будет занимать ровно сто цилиндров, а для того чтобы задать сразу в мегабайтах, необходимо ввести +100M. При этом созданный раздел будет равен ближайшему кратному числу цилиндров. По умолчанию каждому созданному разделу присваивается идентификатор 83.

Более наглядна в работе утилита **cfdisk** (Рис.3), с помощью которой легко проделать с дисковыми разделами все требуемые операции.



Рис. 2

Выбор раздела, с которым будут производиться дальнейшие действия, осуществляется с помощью клавиш вверх-вниз, а выбор требуемого действия – вправо-влево. Все изменения на диск будут записаны только при выборе пункта Write, после чего еще придется для подтверждения ввести yes, до этого можно совершенно не волноваться о сохранности данных. Удалить раздел можно с помощью Delete, пометить как загрузочный – Bootable, Print позволяет просмотреть таблицу разделов или сохранить ее в файле (Рис.4), выбрать тип файловой системы можно с помощью Type, с помощью Units, чтобы не набирать каждый раз размер в мегабайтах, можно сразу установить режим ввода и отображения по умолчанию (Mb, sectors, cylinders), выйти, не внося изменения, – с помощью все той же Quit. Кстати, кто разбивал диски FreeBSD'шной программой /stand/sysinstall, не найдут здесь абсолютно ничего особенного. Можно вместо выбора пунктов меню использовать горячие клавиши, которые совпадают по назначению с таковыми в fdisk, что совсем не вызывает удивление, т.к. вышеописанная утилита является фронт-эндом к ней.

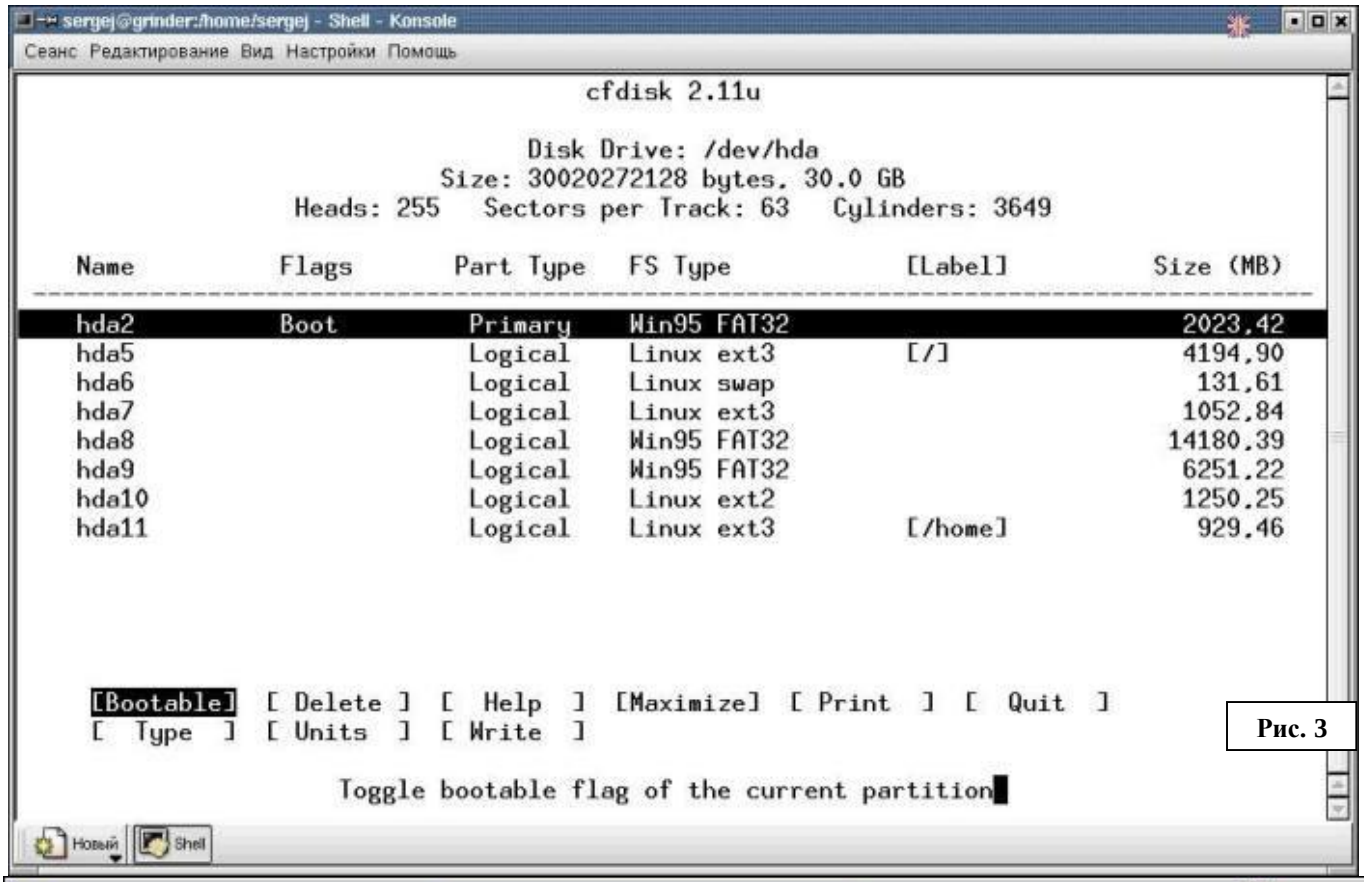


Рис. 3

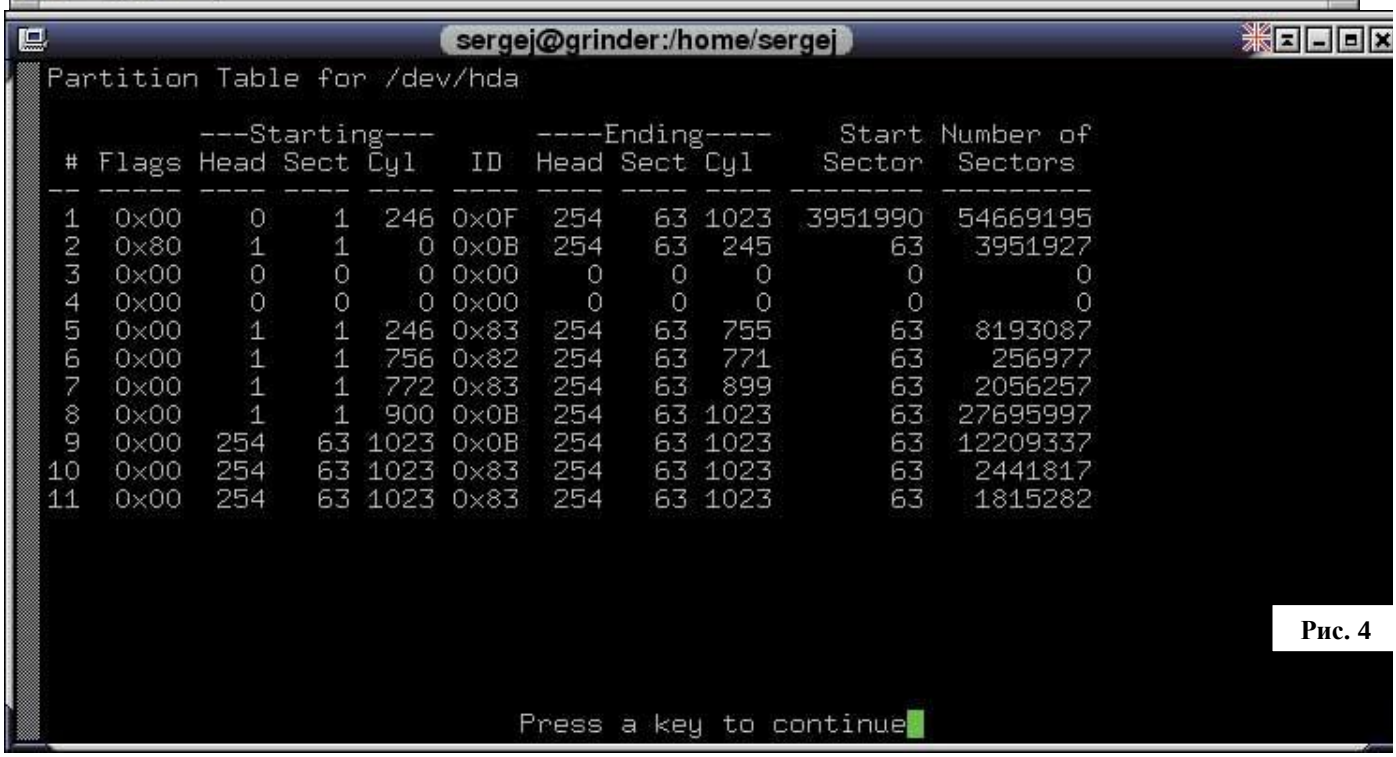


Рис. 4

И наконец, GNU/parted, рожденная в недрах проекта GNU утилита, позволяющая не только создавать новые разделы, но дополнительно и файловые системы на них, к тому же она осуществляет проверку их целостности, а также удаление, перемещение, копирование и, что в новость для Linux, изменение размера разделов – правда, некоторые операции можно производить пока только с ext2fs, swap и FAT16/FAT32.

Работать программа может в двух режимах: в интерактивном и командном.

Первый можно запустить, набрав просто parted в командной строке или добавив опцию -i в командном режиме. После запуска программы и выдачи стандартных для такого рода программ предупреждений и информации о диске, пользователь получает приглашение для ввода команд:

```
(parted)
```

Команда **help** позволяет получить краткую справку, а с аргументом [название команды] – справку по ее работе. Другие команды:

- **print** выводит таблицу разделов;
- **check** MINOR (где MINOR – раздел диска в терминологии parted, который можно узнать с помощью предыдущей команды) производит простую проверку файловой системы;
- **cp** [FROM-DEVICE] FROM-MINOR TO-MINOR – копирует файловую систему в другой раздел;
- **mklabel** ТИП-МЕТКИ – создать новую метку диска (таблицу раздела);
- **mkfs** MINOR ТИП-ФС – создать файловую систему на разделе MINOR; ТИП-ФС может принимать значения ext2, ext3, FAT, hfs, jfs, linux-swap, ntfs, reiserfs, hp-ufs, sun-ufs, xfs; хотя на данный момент поддерживаются только ext2fs, swap и FAT, будем надеяться, что это ненадолго;
- **mkpart** ТИП-РАЗД [ТИП-ФС] НАЧ КОН – создает раздел без файловой системы; ТИП_РАЗД может принимать значения primary, logical, extended. Полезная команда, если раздел удален случайно. НАЧКОН – расстояние в мегабайтах от начала диска;
- **mkpartfs** ТИП-РАЗД ТИП-ФС НАЧ КОН – создать раздел с файловой системой;
- **move** MINOR START [END] – перемещение раздела в пределах диска;
- **name** MINOR NAME – присвоение имени разделу;
- **quit** – выйти из программы
- **resize** MINOR НАЧ КОН – изменить размер файловой системы на разделе, при этом гарантируется (если это вообще возможно при работе с дисками) сохранность данных (размер можно изменить только в соседних разделах);
- **rm** MINOR – удалить раздел MINOR;
- **select** DEVICE – смена рабочего диска;
- **set** MINOR FLAG STATE – изменение флага раздела, где FLAG – boot, root, swap, hidden, raid, lvm или lba, a STATE – on или off.

Взаимодействие с пользователем напоминает bash, тот же механизм автодополнения, редактирования строки ввода, та же история команд. Главное отличие от предыдущих программ – все изменения вступают в силу сразу после ввода команды. Вообще, утилита мне показалась продуманной, логичной и довольно удобной, правда, поначалу несколько непривычной. Но командный режим, скажу вам, это что-то. Введя сразу все необходимые опции, в bash на выходе получим уже готовый размеченный диск:

```
[root@grinder sergej]# parted /dev/hda mkpartfs primary linux-swap 0 128
&& parted /dev/hda mkpartfs primary ext2 129 ###
```

Если необходимо разметить сразу несколько дисков, лучшего инструмента не сыскать. Жаль, правда, что пока возможности ограничены только ext2. Зная мир OpenSource, можно надеяться, что где-то уже пишется фронт-энд к данной утилите, и если это так, то, возможно, мы получим свой бесплатный Partition Magic!

Для работы с разделами можно использовать и программу sfdisk, которая имеет четыре основных режима работы: вывод размера раздела, вывод размера дисков, проверка разделов и еще один, помеченный как very dangerous, - изменение разделов.

Опция s выводит информацию о размерах дисков в блоках:

```
[root@grinder sergej]# /sbin/sfdisk -s
/dev/hda: 29316672
/dev/hdb: 3167640
total: 32484312 blocks
```

Запуск с опцией -l позволяет просмотреть таблицу разделов; если при этом не указывать конкретно диск, то будет выведена информация обо всех дисках (Рис.5). С помощью команды sfdisk -V можно проверить раздел на соответствие записи в таблице и реального состояния; если

все прошло благополучно, то выведет OK, иначе - краткий отчет о проблеме. Создавать разделы с помощью этой программы все-таки неудобно, поэтому трогать мы ее не будем, зато очень легко сохранить таблицу разделов:



Рис. 5

```
[root@grinder sergej]# /sbin/sfdisk /dev/hda -O hda-partition-sectors.save
```

и восстановить ее в случае проблем

```
[root@grinder sergej]# /sbin/sfdisk /dev/hda -I hda-partition-sectors.save
```

Напомню, что эти же операции можно проделать и с помощью команд cat или dd.

Вот мы и рассмотрели все вопросы касательно размещения и создания разделов для ОС Linux на жестком диске.

В прошлой части статьи мы рассмотрели вопросы, связанные так или иначе с дисковыми разделами: их обозначение, рекомендуемое количество и инструменты, предназначенные для их создания. Но создать раздел на диске мало - для того чтобы можно было разместить на нем данные, необходимо позаботиться о создании на нем файловой системы.

Файловые системы: что посеешь, то пожнешь

Под файловой системой понимается, физический способ организации данных на дисковом разделе – возможность их хранения, нахождения и манипулирования ими (запись, стирание). Я думаю, такого простенького определения достаточно, чтобы понять, какие требования выдвигаются к ФС. До недавнего времени в Linux к услугам пользователей предлагалась только одна ФС — ext2fs, предоставлявшая возможность взаимодействовать с ФС других ОС, расположенных на одном диске. Посмотреть перечень таких ФС можно, набрав #make xconfig и зайдя в пункт меню File system. Для включения поддержки их ядром последнее необходимо пересобрать, активировав необходимый пункт. В современных ядрах Linux добавилась возможность работы с различными журналируемыми файловыми системами: ext3fs, ReiserFS, XFS и JFS. Для тех же, кому нужна более гибкая в конфигурировании и быстродействующая файловая система, появилась возможность создавать программные RAID-массивы (раздел raid auto, идентификатор fd) и системы управления логическими томами (LVM, идентификатор 8e). Кроме того, те, кому нужна повышенная конфиденциальность информации, хранимой на компьютере, могут воспользоваться CFS, свободной криптографической файловой системой для Unix/Linux от Матта Блейза (Mutt Blaze). В этой и последующих статьях будут рассмотрены только классические файловые системы, чаще всего применяемые на ПК.

Она была первой: Ext2fs

Как уже говорилось, данная ФС в Linux – это уже стандарт де-факто, ее характеризует довольно высокая надежность и самое высокое из рассматриваемых ФС быстродействие, которое в свою очередь достигается очень эффективным механизмом кэширования дисковых операций. Но так как Linux все чаще используется на высокопроизводительных серверах, то ext2fs уже не удовлетворяет их требованиям – необходимы большие разделы жесткого диска, быстрое восстановление после сбоев, высокопроизводительные операции ввода/вывода, потребность в хранении тысяч и тысяч файлов, представляющих терабайты данных. Все это превышает возможности данной ФС.

Еще одной ее особенностью является тройная косвенная адресация для указания расположения блоков больших файлов. Выглядит это примерно так. Если файл маленький, то в его метаданных содержится прямая ссылка на ячейки, в которых хранятся данные (логические блоки). Это прямая адресация. При увеличении же объема файла до некоторой критической величины занимаемое им место не удается адресовать с помощью прямой адресации. Блоки метаданных указывают уже на косвенные блоки, в которых содержатся адреса с данными, определенными в файле, или опять же указатели на следующие косвенные блоки. Количество таких косвенных блоков для одного файла может достигать трех. То есть если файл увеличивается в размере, внешне это выглядит как единичная операция, внутри же несколько иначе: поначалу распределяются новые блоки, чтобы принять новые данные, затем модифицируется inode, чтобы сделать запись о новых указателях и новых размерах, затем наконец производится запись данных.

Вот теперь представьте, что будет, если при записи файла произойдет сбой. Возможен вариант, когда запись уже произведена или еще не начиналась. Это самый благоприятный вариант: в первом случае после перезагрузки вы так и будете работать с документом, ну а во втором случае потеряется пару часов работы, но с файловой системой ничего страшного не произойдет. А вот если система «упала» именно в момент сохранения файла, то это худшее из того, что могло произойти. Если запись производилась в зону метаданных, то теперь информация, содержащаяся в них, не будет соответствовать реальному расположению файлов на диске. Ситуация усугубляется еще и тем, что Linux, в отличие от Windows, не обязательно записывает обновленный файл поверх старого – при записи во избежание фрагментации выбирается такое место, чтобы он влез полностью. Потому-то в этой системе нет программ-дефрагментаторов (мне доводилось наблюдать фрагментацию данных максимум 0.5%, да и то на переполненном диске, что, согласитесь, очень мало). Так вот, если данные заносились в каталог – а ведь это тоже файл, – то после перезагрузки мы можем недосчитаться одного каталога или, что еще хуже, целого раздела. Ну а если произошел сбой при записи в область данных, то что он будет потом содержать, зависит от вашего везения, особенно в случае, если производилась запись поверх старого варианта файла. Конечно, ситуация не так плачевна, как я обрисовал. За все время активной эксплуатации системы, пережив вместе с ней не одно выключение электропитания, случаев, из ряда вон выходящих, не было (тук-тук-тук).

Естественно, для данной файловой системы (это я все еще о ext2fs веду речь) были разработаны утилиты, помогающие восстановить ее после сбоев. За несколько лет их алгоритм, в отличие, от всеми любимого Scandisk'a, не поленились довести до почти совершенства. Так как проверять при каждой перезагрузке все диски, установленные на компьютере, согласитесь, накладно по времени, нашли такой простой выход. После того как все данные согласованы, непосредственно перед самым ее размонтированием устанавливается cleanbit(в Windows также используется аналогичная технология). Перед загрузкой системы перед ее монтированием программа fsck(FileSystem ChecK) просто проверяет его наличие; если бит установлен, то делается вполне разумное предположение, что файловая система находится в непротиворечивом состоянии, а если нет, то запускается изрядно всем надоевшая утилита fsck (scandisk, по-Microsoft'овски). В связи с тем, что ext2fs содержит избыточные копии критически важных метаданных, вероятность полной потери данных чрезвычайно мала. Система определяет местонахождение поврежденных метаданных и потом либо восстанавливает данные, копируя их из резервных копий, либо просто удаляет файл или файлы, чьи метаданные пострадали. Точнее, не удаляет, а переносит их в

каталог /lost+found. В этом-то и состоит очевидное неудобство: согласитесь, что чем больше файловая система, тем дольше длится процесс проверки. На дисковом разделе размером в несколько гигабайт, что давно уже перестало быть редкостью, с большим количеством файлов и каталогов, процедура проверки метаданных во время загрузки может очень сильно затянуться. А если выбило главный производственный сервер, и теперь пользователи вынуждены ждать, пока он перегрузится? Вот так мы плавно подошли к журналируемым файловым системам.

Что есть журнал?

Вся волшебная сила журнала заключена в механизме транкзаций – этот термин неплохо известен тем, кто работал с базами данных. Как и там, механизм транкзаций вместо отслеживания модификаций к таблицам и данным, рассматривает операцию записи на диск как атомарную, а не разделенную на несколько этапов, что позволяет отследить, прошла ли запись вообще, и в свою очередь гарантировать, что изменение файловой системы произведено полностью или не произведено вообще. Поясню сказанное на примере. Например, при создании нового файла изменяются несколько структур метаданных (inodes, списки свободных блоков, список файлов каталога идр.) Прежде, чем в файловой системе произойдут изменения, создается транзакция, в которой описывается то, что должно быть сделано. Как только транзакция будет зарегистрирована (на диске), файловая система приступает непосредственно к изменению метаданных. То есть фактически журнал в такой файловой системе – просто список производимых операций. В случае системного сбоя файловая система будет восстановлена к непротиворечивому состоянию путем повторного запуска журнала и отката к предыдущему состоянию. К тому же в таком случае файловая система осматривает только те участки диска, в которых изменялись метаданные, т.е. она уже «знает», где произошел сбой. Это намного быстрее, чем при традиционной проверке с помощью fsck. И что самое существенное, время восстановления совсем не зависит от размера раздела – скорее, от интенсивности операций на момент сбоя.

Можно выделить два возможных варианта работы журналируемой файловой системы. В первом варианте в журнал заносятся только изменяемые метаданные файла, в таком случае при сбое будет гарантированы метаструктуры файловой системы, а сохранность самих данных уже зависит от везучести. Второй вариант предусматривает занесение в журнал вместе с метаданными также и самих данных, как изменившихся, так и не модифицированных, в этом случае есть вероятность, что данные после сбоя будут восстановлены. И конечно же, как говорил мой преподаватель по теоретическим основам электроцепей, «природу не обманешь, за все нужно платить», а платить теперь приходиться быстродействием, так как самые медленные операции в компьютере – это операции ввода/вывода на диск, а количество таких операций возросло, особенно при использовании варианта с журналированием данных. Решают вопрос разными ухищрениями: например, запись происходит в момент наименьшей активности, некоторые журналируемые файловые системы позволяют разместить журнал на другом физическом диске. Да и фактически время работы с журналом намного меньше, чем работа непосредственно с данными. И естественно, некоторый полезный объем теперь приходится отводить под сам журнал, но его размеры как правило не превышают 32Мб, что по нынешним временам не так уж и много.

Самое главное, что вы должны запомнить: журналируемые файловые системы предназначены не для восстановления всех ваших данных любой ценой, главная их задача заключается в поддержании непротиворечивости метаданных файловой системы на момент сбоя.

Большинство современных journaling файловых систем поддерживают:

- более быстрое распределение свободных блоков; для этого большинство из них построено на основе сбалансированных деревьев, иначе известных как B+деревья. О том, что это такое, лучше спросите у авторов, пишущих о различных языках программирования, особенно об алгоритмах поиска и сравнения, а особо любопытные пусть посмотрят документ по адресу http://starship.python.net/crew/aaron_watters/bplustree/bplustree.py.txt;
- большее количество файлов в каталоге: поскольку обычная связка name-inode становится неэффективной, то опять же для хранения имен файлов используются B+деревья. В некоторых случаях возможно использование всего одного B+дерева для полной системы,

что намного укорачивает поиск файла и, соответственно, операции по работе с ним. Плюс динамическое выделение inides вместо неэффективного статического;

- старая методика прямого, косвенного т.д. механизма хранения информации о нахождении данных файла, очень неудобная при работе с файлами большого размера по причине долгого поиска информации, заменена на более гибкую, позволяющую работать с большими файлами «напрямую»;
- кроме того, некоторые новые файловые системы имеют более совершенный механизм управления внутренней фрагментацией (что это такое, объясню чуть ниже) и распределения inodes, чем Ext2. Может, конечно, сложиться впечатление, что место журналируемым файловым системам – где-нибудь на сервере; нет, они подходят на все сто процентов для использования на клиентских машинах, везде где есть необходимость в сохранении данных. Теперь, когда мы точно знаем, что ожидать от описываемых файловых систем, перейдем к их конкретной реализации.

Система в красной шапке: Ext3fs

Хотя данная файловая система не была первой, поддерживаемой ядром Linux официально (появилась только с версии 2.4.16), все таки я думаю, что справедливо будет начать именно с нее. Разработана она в недрах компании RedHat (там и следует искать всю информацию о ее работе) доктором Стефеном Твиди (Stephen Tweedie). Найти патчи для ядра можно по адресу ftp://ftp.linux.org.uk/pub/linux/sct/fs/jfs. Чтобы не изобретать колесо, поступили просто, прикрутив к стандартной ext2fs журнал (в зависимости от опций монтирования, его можно и не видеть; находится в ./.journal) и заменив драйвер ядра, отвечающий за файловую систему. По этой причине она, естественно, наследует все достоинства и недостатки, присущие ext2fs. Но что это дало?

Самое главное – это то, что утилиты ext2fs, которые шлифовались в течение нескольких лет, работают в ней как ни в чем не бывало. К тому же идентичность файловых систем позволяет оперативно переходить как с ext3fs на ext2fs, так и наоборот. Поясню. Мне часто приходится устанавливать другие дистрибутивы, в том числе и со старыми ядрами, не поддерживающими новинку. Так вот, все разделы, на которых используется ext3fs, я монтирую просто как ext2fs – и никаких, повторяю, никаких недоразумений при использовании не происходит.

Другое преимущество данной файловой системы состоит в том, что она, в отличие от остальных, поддерживает режим журналирования данных (полное или частичное). Естественно, добавление журнала, казалось бы, должно было ухудшить производительность такой системы по сравнению с «нежурнальным» вариантом. Оказалось, что за счет улучшения алгоритма движения головки жесткого диска данная файловая система в некоторых случаях даже обходит ext2fs. Ext3fs имеет три режима работы:

- data=writeback – режим, при котором не выполняется никакого журналирования данных, учитываются только метаданные – самый ограниченный режим журналирования (кстати, применяемый во всех других ФС рассматриваемых ниже), не гарантирующий сохранности данных после сбоя. Но за счет этого возрастает скорость работы такой файловой системы: фактически журнал предназначен только для того, чтобы уменьшить время начальной загрузки системы;
- по умолчанию же используется data=ordered – золотая середина между полным журналированием данных и предыдущим режимом. Официально в этом случае журналируются только метаданные, но блоки соответствующих им данных записываются первыми. В большинстве случаев такой режим гарантирует сохранность данных, особенно если данные дописывались в конец файла, чего в большинстве случаев предостаточно. Производительность, естественно, чуть ниже предыдущей и выше режима полного журналирования – data=journal, – в котором все новые данные сначала пишутся в журнал и только после этого переносятся на свое законное место. В случае аварийного отказа журнал можно повторно перечитать, приведя данные и метаданные в непротиворечивое состояние.

Кстати, как оказалось, данный режим в случае, когда диск интенсивно загружен операциями IO, оказывается даже быстрее всех остальных.

Выбрав режим, отличный от установленного по умолчанию, необходимо указать его с помощью опции -o. Например:

```
#mount -o data=journal -t ext3 /dev/hda5 /usr/local
```

Или в /etc/fstab:

```
/dev/hda5 /usr/local ext3 data=writeback 1 0
```

или если режим по-умолчанию, то просто

```
/dev/hda5 /usr/local ext3 data=writeback 1 0
```

Если же теперь захочется отказаться от него, то, исправив ext3 на ext2, можно забыть о журнале:

```
/dev/hda5 /usr/local ext2 data=writeback 1 0
```

Для того чтобы к обычной ext2fs добавить журнал, достаточно выполнить команду

```
# /sbin/tune2fs -j /dev/hda5
```

причем на неразмонтированной файловой системе. При этом вроде как гарантируется сохранность данных, хоть предварительно заархивировать их все же не повредит. Для того чтобы изначально создать ext3 на пустом только что созданном разделе диска, достаточно выполнить команду:

```
# /sbin/mke2fs -j /dev/hdb5
```

Начиная с версии 0.9.5, ext3fs может использовать другой диск для хранения файла журнала.

Подробности можно узнать по адресу http://www.zipworld.com.au/~akpm/linux/ext3/ext3-usage.html.

Вот и все о данной файловой системе. От себя хочу заметить, что разделы /home, /usr/local, которые часто приходится монтировать к другим Linuxам, у меня отформатированы именно под ext3. Что и говорить, это предсказуемая и, главное, УДОБНАЯ файловая система. Характеристики относительно максимального количества файлов и каталогов, а также максимальных размеров разделов меня полностью устраивают. Но у нее есть один наследственный недостаток, который практически полностью решен в другой ФС. Но об этом поговорим чуть позже.

ReiserFS

Это первая «сторонняя» файловая система, появившаяся в официальном ядре 2.4.4. На первое время ее работа вызывала одни только нарекания, поэтому ее использовали только любители острых ощущений. Данный проект стартовал в 90-х годах, первый прототип носил название TreeFS. Разработана Хансом Райзером (Hans Reiser) и его компанией Namesys (http://www.namesys.com), причем задачи они перед собой поставили очень, я бы сказал, революционные. Разработчики системы мечтают создать не только файловую систему, но вообще механизм иерархического именования объектов. Они считают, что лучшая файловая система та, которая формирует единую общедоступную среду – namespace. Для этого файловая система должна выполнять часть работы, традиционно выполнявшуюся приложениями, что уменьшит количество несовместимых API узкоспециального назначения. При таком подходе пользователи смогут продолжать прямое использование файловой системы без необходимости формировать уровни специального назначения, типа баз данных и т.п. А вообще, зайдите на сайт, благо из всех подобных проектов этот наилучшим образом документирован. Но предупреждаю, документации там много. Базируется она на оптимизированных b*–сбалансированных деревьях (одно на файловую систему), использование которых кроме увеличения производительности снимает ограничение на количество каталогов (хотите 100 тыс. – без проблем!) На данный момент поддерживает журналирование только метаданных, но разработчики обещают в скором времени предусмотреть и режим, аналогичный data=journal в ext3. Преимущества данной ФС в основном проявляются в работе с маленькими файлами. Поясняю. Как я уже говорил, информация на физическом носителе хранится не как попало, а отдельными блоками, размер которых зависит и от размера раздела (это связано с максимально возможной адресацией) в том числе (устанавливается при форматировании), но в большинстве случаев равен 4 Кб. Так вот, ext2 (ext3 и FAT тоже) могут адресовать только целое количество блоков. Ну и что? Имеем файл 10 Кб, размер блока 4 Кб. Получается, что файл займет 2 целых блока и один только на половину – 4+4+2 (2 осталось незанятыми, это и называется внутренней фрагментацией). Для единичного файла это не страшно,

но если их несколько тысяч? По подсчетам, в этих ФС теряется где-то от 6 до 10 процентов. Кстати, Fast File System (FFS), применяемая в BSD, умеет адресовать субблоки, а во всеми любимой FAT придется мириться с неизбежной потерей места. Разработчики ReiserFS решили отказаться от решения вереницы противоречивых задач, сосредоточившись на одной. ReiserFS может запросто упаковывать несколько маленьких файлов в один дисковый блок (tail packing), а совсем маленькие вообще просто запихать в inode (внутрь b*tree). По необходимости для файла может ассигноваться точный размер. Такой режим работы предусмотрен по умолчанию, но для повышения быстродействия есть возможность ее отключить. Хотя показатели ReiserFS при работе с большими файлами довольно высоки, именно работа с маленькими файлами (меньше Кб) и обслуживание большого их количества выделяет данную ФС. По работе с ними она превосходит по быстродействию все представленные файловые системы (видел цифры: в 8-15 раз), именно за счет того, что данные и метаданные хранятся рядом, но с «разреженными» файлами работает все-таки хуже (это, как ожидается, будет исправлено в четвертой версии, ожидаемой 30 июня 2003 года). Плюс, как видите, достигается значительная экономия дискового пространства. Различные источники называют ReiserFS самой устойчивой из всех рассматриваемых ФС; ее, я думаю, можно рекомендовать для корневого раздела, который к тому же состоит из маленьких файлов. Такая себе рабочая лошадка. Но для работы с данной ФС, кроме поддержки ее самим ядром, необходимы также специфические утилиты для работы и обслуживания разделов – они уже входят в стандартную поставку всех современных дистрибутивов, а если нет, то можете взять их по адресу ftp://ftp.namesys.com/pub/reiserfsprogs/reiserfsprogs-3.6.4.tar.gz.

Если ядро уже поддерживает ReiserFS и имеются необходимые утилиты, то набрав

```
# /sbin/mkreiserfs /dev/hda2
```

можно создать на ней соответствующую файловую систему. Для автоматического монтирования ее при загрузке достаточно прописать в файле /etc/fstab

```
/dev/hda2  /home     reiserfs  defaults  0 0
```

или

```
#/sbin/mount -t /reiserfs dev/hda2 /home
```

при монтировании вручную. Если для увеличения производительности необходимо отключить упаковку хвостов, то добавьте опцию notail:

```
/dev/hda2  /home     reiserfs   notail 0 0
```

А опция -genericread может увеличить производительность при операциях поиска файлов, т.е. когда головка мало считывает, но много перемещается по диску.

XFS

Основа этой файловой системы была создана в начале 90-х (1992-1993) фирмой Silicon Graphics (сейчас SGI) – чувствую, как напряглись те, кто занимается графикой, — для мультимедийных компьютеров с ОС Irix, заменив уже не удовлетворявшую требованиям времени EFS, но немного «очищенная» версия 1.0 стала доступна только первого мая 2001. Найти все необходимую информацию можно по адресу http://oss.sgi.com/projects/xfs. Файловая система была ориентирована на ну очень большие файлы (9 тыс. петабайт – 9 млн. терабайт – 1018 байт) и файловые системы (18 тыс. петабайт) – в отличие от предыдущих, она является полностью 64-битной, что позволяет адресовать большие массивы данных. Особенностью этой файловой системы является устройство журнала – в журнал пишется часть метаданных самой файловой системы таким образом, что весь процесс восстановления сводится к копированию этих данных из журнала в файловую систему. Размер журнала задается при создании системы, он должен быть не меньше 32 мегабайт (больше, наверное, и не надо – такое количество незакрытых транзакций тяжело получить). Тесты на производительность показывают бесспорное преимущество XFS, особенно при работе с большими и средними файлами.

Также эту файловую систему характеризует прямолинейность падения производительности при увеличении нагрузки и предсказуемость - дополнительно она не генерирует излишнюю дисковую активность, т.к. пытается кэшировать как можно больше данных и «основанием» для сброса на диск является заполнение памяти, а не интервал времени, как это принято в других ФС. Любое дисковое устройство при создании файловой системы XFS разбивается на несколько

равных по размеру линейных областей (0.5-4 Гб), в терминологии XFS они именуются allocation group. Уникальность allocation group в том, что каждая группа управляет своими собственными inodes и свободным местом, что превращает группы в своего рода автономные файловые системы, сосуществующие в рамках общей XFS. Такая организация позволяет эффективно организовать параллельную обработку операций ввода/вывода, которая особенно ярко проявляется на многопроцессорных системах. В каждой такой группе используется три B+-дерева, два из которых отвечают за свободные inodes (allocation). В этой системе реализована очень хорошая возможность, позволяющая избежать фрагментации файлов, называемая delayed allocation.

ТАБЛИЦА

	Ext3	ReiserFS	XFS	JFS
Самый большой размер блока на ia32	4 Кб	4 Кб	4 Кб	4 Кб
Максимальный размер файловой системы	16384 Гб	17592 Гб	18000 Пб+	32 Пб
Максимальный размер файла	2048 Гб	1 Эб*	9000 Пб	4 Пб
Размещение журнала на внешнем устройстве	Да	Да	Да	Да
Журналирование данных	Да	Нет	Нет	Нет
Поддержка Dynamic disk inode allocation	Нет	Да	Да	Да
Поддержка Access Control Lists	Patch	Нет	Да	WIP
Red Hat 7.3	Да	Да	Нет	Да
SuSE 8	Да	Да	Да	Да
Mandrake Linux 8.2	Да	Да	Да	Да
Slackware Linux 8.1	Да	Да	Да	Да

+ Пб это 10^{15} байт * Эб это 10^{18} байт
Но ядра серии 2.4.x имеют предел 2048 Гб обойти его можно, установив патч.

При этом файловая система, получая данные для записи, по началу лишь резервирует под них необходимое свободное место, откладывая саму запись до момента фактического сброса данных. Когда же такой момент наступает, XFS решает, куда необходимо их поместить. Если осуществляется дозапись, то подбираются соседние сектора. Но наибольший эффект от такой задержки получается еще и за счет того, что при создании временного файла с малым временем жизни последний вообще на диск не пишется (соответственно, не приходится занимать/освобождать метаданные). Для борьбы с внешней фрагментацией (это как раз то, против чего борются программы типа Norton Speed Disk) разработчики в ближайшее время планируют выпустить аналогичную утилиту. К сожалению, каноническим ядром пока данная ФС не поддерживается, хотя в экспериментальных 2.5.x версиях ядра поддержка ее уже включена, что позволяет надеяться на скорое решение этого вопроса, а некоторых дистрибутивах (Gentoo, Lunar Linux) она уже предлагается пользователю. Так что придется сходить на сайт разработчика за патчем (ftp://oss.sgi.com/projects/xfs/download) и необходимыми утилитами (как минимум xfsprogs) для работы с ней. Сейчас на сайте доступен релиз 1.2pre4, меньше 1.1 брать точно не стоит, в них были замечены некоторые ошибки, в частности, малая скорость удаления большого количества файлов. Теперь, пересобрав ядро и установив необходимые утилиты, можно создать файловую систему:

```
#/sbin/mkfs.xfs /dev/hdb2 или mkfs -t xfs /dev/hdb2
```

Для увеличения производительности в некоторых случаях может помочь опция -l size=32m, фиксирующая размер журнала (32 Мб), также с помощью -d agcount=x хорошо бы установить минимально возможное количество allocation groups (т.е. взяв максимально возможные 4 Гб на группу). Например, при разделе 18 Гб устанавливаем:

```
#/sbin/mkfs.xfs -d agcount=5 -l size=32m -f /dev/hdb2
```

Необязательная опция -f позволяет создать XFS поверх любой существующей ФС, но при создании раздела поверх ReiserFS (и наоборот) необходимо заполнить нулями начальный раздел, содержащий метаданные перед переформатированием, т.к. команда mount может неправильно определить, какая из файловых систем установлена. Вот как это делается:

```
# dd if=/dev/zero of=/dev/hdb2
```

Прервать операцию секунд через 10 - 20 комбинацией Ctr+C. Смонтировать вновь созданный раздел теперь можно командой

```
# mount -t xfs =/dev/hdb2 /home или в файле /etc/fstab
/dev/hdb2  /home     xfs  defaults  0 0
```

Для повышения производительности можно задать некоторые опции noatime, nodiratime, osyncisdsync, вместе помогающие добиться асинхронного вывода информации и практически имитировать поведение ext2, а также logbufs, устанавливающую размер буфера (по умолчанию равен 2), – здесь особо усердствовать не стоит, например, 8 при 128 Мб оперативной памяти уже многовато:

```
/dev/hdb2  /home    xfs  noatime, nodiratime, osyncisdsync,logbufs=4   0 0
```

Остальную информацию смотрите в каталоге /usr/src/linux/Documentation/filesystems, файл xfs.txt.

JFS (Journaled File System)

Первоначально создана фирмой IBM для своей OS/2 Warp, а затем выпущена по лицензии GPL и портирована под Linux. Всю необходимую информацию можно получить по адресу http://oss.software.ibm.com/jfs. По своим характеристикам и архитектуре очень схожа с предыдущей, поэтому вдаваться в подробности не буду. Как и в предыдущей, в этой файловой системе раздел логически подразделяется на «агрегаты», но последние включают, кроме данных, еще и отдельный журнал, при этом каждый из таких сегментов можно монтировать отдельно; также имеется возможность хранения маленьких файлов в пределах inode. Если катлог имеет до 8 файлов, то информация о них содержится в самом inode, при увеличении же их количества используются уже знакомые B+ –деревья. По тестам это, наверное, самая медленная файловая система из рассматриваемых, хотя и разрабатывалась она для работы на высокопроизводительных серверах. Для установки необходима утилита jfsutils, патч к ядру jfs-2.4.x-patch и код ФС jfs-2.4-1.0.20.tar.gz. После установки и компиляции всех программ для создания раздела достаточно выполнить команду

```
# mkfs.jfs /dev/hdb3
```

и смонтировать ее:

```
# mount -t jfs /dev/hdb3 /jfs
```

Для возможности работы с внешним журналом необходимо два неиспользуемых раздела, например:

```
# mkfs.jfs -j /dev/hdb1 /dev/hda6
# mount -t jfs /dev/hda6 /jfs или в /etc/fstab
/dev/hda6 /jfs jfs defaults 1 2
```

Как видите, ОС Linux предоставляет пользователю возможность выбрать даже файловую систему под конкретные задачи. И самое главное, необязательно, чтобы была установлена только одна из этих файловых систем. Например, для корневого раздела вполне подойдет ReiserFS, для /usr/local — ext3, а домашний каталог, битком набитый видео, можно отформатировать под XFS.

Каждому свой крест

Сергей ЯРЕМЧУК, grinder@ua.fm

Хорошо живется нашему брату линуксоиду. Дистрибутивов в природе много, не нравится один – пожалуйста, всегда есть из чего выбирать. Не нравится любой из имеющихся – тоже не беда, наличие свободно доступного кода любого из компонентов системы, подробного

руководства по созданию своего дистрибутива ***Linux From Scratch*** *(http://www.linuxfromscratch.org, русский перевод на http://multilinux.sakh.com/lfs) позволяет создать дистрибутив сообразно своему видению и представлению о том, каким он должен быть. После этого, правда, уже некого будет ругать за неудобство программы установки, за базовый состав программ. Все в ваших руках. И что самое интересное, создают ведь. На таких сайтах как http://www.linux.org/dist и* ***DistroWatch*** *(http://www.distrowatch.com) ссылки на новые дистрибутивы появляются чуть ли не каждую неделю. Уследить за этим процессом, по-моему, - дело совершенно бесполезное. Вот так и мне как-то попалось сообщение о новом проекте, особенностью которого была система портов, аналогичная* ***FreeBSD****, но зайдя на сайт, я увидел, что номер версии еще далек от релиза, и решил подождать. И дождался таки. В декабре прошедшего года наконец вышла версия 1. Итак, встречайте* ***CRUX****.*

Легкий i686-оптимизированный дистрибутив, рассчитанный на подготовленного пользователя (Power user). Да и названьице подобрано с явным умыслом, чтобы всем было ясно, для кого он предназначен. Разработчикам, наверное, надоели переполненные в большинстве своем ненужными пакетами дистрибутивы, и они пошли по проторенной Unix дорожке. KISS (keep it simple stupid) – таков девиз UNIX, таков же девиз CRUX. Простота достигается удобной системой, базирующейся на tar.gz-пакетах, BSD-стилем загрузочных скриптов и небольшим количеством первоначально устанавливаемых приложений. В итоге вместе с системой X-Window можно запросто уложиться в 500 Мб (и даже меньше), причем это будет самая современная система, а не какой-нибудь RH 6. Итак, кто не испугался, вперед… На сайт проекта http://crux.nu. Там с радостью узнаем, что для пользователей более слабых i586-совместимых тоже не все потеряно – найдется версия и для их компьютеров. Только внимательно смотрите, что скачиваете. ISO-образ занимает 188 Мб, для модема, конечно, многовато, но через выделенку 256 кб/с с утра в воскресенье, когда каналы еще разгружены, за четыре часа он перебрался ко мне на винт. Дополнительно советую также скачать **CRUX Handbook**, там найдете ответы на множество вопросов, возникающих на первоначальном этапе ознакомления с этой системой.

Для начала проверим образ на успешность загрузки с помощью контрольной суммы, при наличии такой возможности, конечно:

```
$ md5sum crux-1.0.iso
91f2e0d8febe2d3c25cf66e801376178
```

Сравним ее с написанной в файле crux-1.0.md5sum, лежащим тут же на сайте. Тем самым мы получаем гарантию, что образ загружен нормально и что никто не подменил его на сайте (вообще, сравнение контрольной суммы должно войти в привычку). Далее записываем образ на диск, благо проблем с этим сейчас нет, вставляем его в CD-ROM и перезагружаем компьютер, не забыв изменить соответствующие настройки BIOS. Желательно также установить BIOS Virus Protection в DISABLED, иначе могут возникнуть проблемы при записи новой таблицы разделов на диск. Появляется заставка с белесой надписью CRUX и приглашение к загрузке boot:. Написано, что для справки можно нажать F1 – можете не тратить время, там ничего пока интересного нет. Просто жмем Enter и ждем, пока загрузится ядро. После чего нас просят ввести логин, набираем root и без пароля попадаем прямиком в shell. Так что любители графических установок здесь отдыхают – никакого интерфейса, за исключением командной строки (правда, полноценного bash, с историей команд, автодополнением и прочими приятностями), в том числе и программы установки, как в Lunar Linux, не предусмотрено. Но не все так мрачно. У меня весь процесс установки занял времени не более 30 мин. Параллельно доступно еще 6 виртуальных консолей, так что жить можно. На второй консоли, куда переходим по Alt+F2, вводим:

```
$ less /crux/install.txt
```

И сверяем свои дальнейшие шаги по этому документу. Далее нам предлагается вручную создать необходимые разделы и файловые системы на них. В ядре системы реализована поддержка файловой системы управления устройствами devfs; все обозначения дисковых разделов в CRUX приняты исходя из этого. В указанном файле install.txt предлагается ввести команду

```
$ fdisk /dev/discs/disc?/disc
```

и далее работать с этой программой. Попробовал ввести

```
$ cfdisk /dev/discs/disc?/disc
```

Знаете, получилось – так что кто с первой не дружит, можете создавать разделы в более понятной среде (GNU/parted нет). Под систему желательно отвести с учетом будущего расширения хотя бы 1 Гб, но можно вложиться, в зависимости от выбранных в дальнейшем программ, и в 200 Мб. Следующим шагом будет создание файловых систем на только что созданных разделах. Для корневого раздела создатели CRUX рекомендуют ReiserFS, я также придерживаюсь этого мнения. Поэтому набираем:

```
$ mkreiserfs /dev/discs/disc?/part?
```

И для создания своп-раздела, если его еще нет на диске:

```
$ mkswap /dev/discs/disc?/part?
```

Сразу же активизируем своп-раздел:

```
$ swapon /dev/discs/disc?/part?
```

Монтируем созданный раздел под корневую систему:

```
$ mount /dev/discs/disc?/part? /mnt
```

Если дополнительно были созданы разделы для каталога /home и других, то поступаем аналогично:

```
$ mkdir /mnt/home
$ mount /dev/discs/disc?/part? /mnt/home
```

После того как созданы все необходимые разделы будущей системы и примонтированы каждый в свое место, приступаем собственно к установке. Для этого набираем Setup. После чего появляется заставка с двумя пунктами: начать новую CRUX-инсталляцию и проапгрейдить имеющуюся систему. Выбираем нужное и дальше указываем точку монтирования созданной корневой системы. По умолчанию предлагается /mnt. Приступаем к выбору пакетов для установки. Все они разделены на две группы: base – рекомендуется установить все пакеты из этой группы; и opt – как следует из названия, опциональные пакеты, куда входят X-Windows и необязательные прочие программы. При этом на данный момент не реализовано никакой проверки зависимостей, но с другой стороны, вам и не навалят то, что не просили. Далее идет непосредственно установка пакетов, которая на моем слабеньком компьютере не превысила 20 минут. Теперь, для того чтобы созданные файловые системы автоматически загружались при старте, необходимо подправить файл /mnt/etc/fstab. Для этого в системе есть vi и более понятный для новичка pico. Вызываем файл на редактирование:

```
$ pico /mnt/etc/fstab
```

Хочется отметить, что разразработчики не стали усложнять жизнь, а заготовили шаблоны. Например:

```
# /dev/#REISERFS_ROOT# / reiserfs defaults 0 0
```

Причем, такие строки существуют для всех поддерживаемых ядром файловых систем. Остается только раскомментировать нужную строку, и вместо #REISERFS_ROOT# вписать нужный раздел.

Теперь пришла пора рассказать, что же за файловые системы поддерживает дистрибутив. Кроме уже упомянутой Reiser FS и традиционных ext2fs/ext3fs, vfat, iso9660 (ФС CD-ROM), ядро поддерживает еще и JFS, а также tmpfs, которую предлагается монтировать либо в /tmp, либо в /dev/shm (я выбрал первый вариант – так легче избавляться от временных файлов), и, конечно же, UDF (Universal Disk Format) – файловая система, применяемая в DVD-дисках и позволяющая использовать CD-R и CD-RW диски как обыкновенную дискету. Дополнительно на сайте уже появилась информация о поддержке XFS, так что выбирать есть из чего. После того как все необходимые разделы будут прописаны, нажимаем Ctrl+X для выхода и соглашаемся с тем, что файл нужно сохранить. Следующим шагом будет редактирование /mnt/etc/rc.conf, в котором необходимо указать клавиатурную раскладку (можно посмотреть имеющиеся в /mnt/usr/share/kbd/keymaps/i386/qwerty), установить часовой пояс (/mnt/usr/share/zoneinfo/) и указать автоматически запускающиеся при старте системы сервисы. У меня он получился такой.

```
KEYMAP=ru4
TIMEZONE=Europe/Kiev
HOSTNAME=my_host
SERVICES=(net crond)
```

Обратите внимание на строку SERVICES, в которой просто перечислены сервисы, которые необходимо автоматически запустить при старте системы; все имеющиеся на данный момент

можно найти в /mnt/etc/rc.d/. Далее рекомендуется подправить /mnt/etc/rc.d/net, /mnt/etc/hosts и /mnt/etc/resolv.conf для настройки сетевых соединений, но я думаю, пока этот этап можно пропустить. В принципе, на этом можно было установку и закончить, если бы не одно маленькое «но». Все, чем мы занимались до этого, особых проблем для новичка не представляет. Наибольший же страх в Linux всегда вызывала... компиляция ядра. Если внимательно посмотреть в каталог /boot, то обнаруживается, что там совершенно пусто. Итак, вперед к кошмарам!

Сначала перезагружаемся и загружаемся опять с CD-ROM, но теперь при приглашении boot: вводим CRUX root=/dev/discs/disc?/part?, чтобы смонтировать созданный раздел как корневой. Внимательно просмотрите сообщения об ошибках, обращая внимание на монтируемые файловые системы. Далее обычная :-) сборка ядра. Переходим в /usr/src/linux-2.4.20 и последовательно выполняем команды:

```
$ make menuconfig
```

Здесь необходимо для включения поддержки devfs в ядре обязательно включить опции в

```
Code maturity level options --->
[*] Prompt for development and/or incomplete code/drivers
```

А также в

```
File systems --->
[*] /dev file system support
[*] Automatically mount at boot
```

И конечно же, не забудьте включить те файловые системы, под которые отформатированы ваши разделы, и установить тип процессора, соответствующий вашему, иначе ничего не получится. После всего этого выходим из программы конфигурирования и по запросу сохраняем результат. Набираем

```
$ make dep && make bzImage
```

Копируем ядро на положенное ему место:

```
$ cp /usr/src/linux-2.4.20/arch/i386/boot/bzImage /boot/vmlinux
```

И если собиралось модульное ядро, то дополнительно вводим:

```
$ make modules
$ make modules-install
```

Если не было никаких сообщений об ошибках, то правим файл /etc/lilo.conf, для загрузки ядра устанавливаем lilo -v и перезагружаемся уже в нормальном режиме, т.е. с жесткого диска. Ждать, когда на моем компьютере скомпилируется ядро, совершенно нет мочи. Поэтому я обычно компилирую их на ночь глядя, записываю все команды, включая копирование, в файл, добавляю /sbin/shutdown -h now, делаю его исполняемым (chmod +x) и запускаю на выполнение. С утра загружаюсь уже с новым ядром.

Самое время рассказать, что же нам предлагают из компонентов:

- ядро Linux 2.4.20;
- GNU C Library 2.3.1;
- GCC 3.2.1;
- Binutils 2.13.1;
- Perl 5.8;
- XFree86 4.2.1;
- Sendmail 8.12.6;
- OpenSSH 3.4;
- Vim 6.1;
- оконный менеджер WindowMaker 0.80.2

Есть и XMMS, Mozilla 1.2 и консольные утилиты для работы с CD-R дисками, но удивило отсутствие консольных браузеров Links или хотя бы Lynx – может, графический режим считается создателями основной рабочей средой? Но как видите, все самое свежее – для этого, собственно говоря, CRUX и создавался, чтобы всегда держать систему в самом современном состоянии. Нашелся в коллекции устанавливаемых утилит и забавный скрипт, называемый rpm2targz, предназначенный для распаковки rpm-пакетов в архивы (работает – аж шуршит). Я даже скопировал его вместе с необходимой для работы библиотекой, находящейся в каталоге

/usr/lib/rpm2targz, в мой дежурный Red Hat, изменив при этом пути в самом скрипте, чтобы положить в /usr/local/lib.

Теперь можно заняться установкой пароля, завести пользователя, под которым будем работать постоянно, а также озаботиться настройкой сети и X-Window. Для настройки X-Window достаем документы с характеристиками монитора и вводим xf86config для запуска программы или xf86cfg для настройки с помощью графической программы (если получится). Несколько огорчает, что нет русской и украинской локали - конечно, и соответствующих шрифтов также нет, так что впору доставать FAQ'и конца прошлого столетия, там подробно написано, как локализовать дистрибутив (посмотрите на http://www.sensi.org/~ak/linuxfaq/rulinux.faq.html). По умолчанию стартует twm; чтобы сменить его на более удобный WindowMaker, самым простым вариантом будет заменить в файле /etc/X11/xinit/xinitrc соответствующую строку (twm –> wmaker).

Таблица 1

Уровни запуска	***RedHat***	***CRUX***
0	останов (halt)	останов (halt)
1	однопользовательский режим	однопользовательский режим
2	многопользовательский режим без NFS (то же, что и 3, но без сети)	полностью многопользовательский режим
3	полностью многопользовательский режим	Не используется
4	Не используется	Не используется
5	X-Window	Не используется
6	Перезагрузка (reboot)	Перезагрузка (reboot)

Как я говорил, в дистрибутиве применен более BSD'шный стиль загрузочных скриптов. И действительно, посмотрев в файл /etc/inittab, обнаруживаем, что уровни запуска совсем другие, чем в тех же Red-Hat based дистрибутивах (см. Табл. 1). Как видите, запустить автоматически стандартным путем не получится, а запускать напрямую xdm (сервис, предназначенный для регистрации пользователя в графическом режиме) значит лишить себя возможности перейти в другую консоль, если с X-Window что-то не так (такая вероятность все же существует). Но чтобы такой режим вообще не был предусмотрен, мне не верилось. В каталоге /etc/rc.d/ обнаружился забавный скрипт xdm, который и управляет загрузкой данного сервиса. Так что добавляем в файле /mnt/etc/rc.conf в переменную SERVICES значение xdm и радуемся автоматической загрузке X-Window.

Отдельно хочу пару слов замолвить о системе портов и пакаджей, применяемой в CRUX. Устанавливаются порты так же, как и во FreeBSD - $ pkgadd xmms-1.2.7-1.pkg.tar.gz, при этом предварительно скачанный с сайта порт установится, дополнительно при этом будет произведена проверка зависимостей. Если пакет уже установлен, но требуется обновить его, то, набрав $ pkgadd -u package_name, избежим сообщения о том, что пакет уже установлен. Причем, чтобы предотвратить модификацию некоторых важных для системы каталогом и файлов в /etc/pkgadd.conf, описываются правила. Например, строки:

```
UPGRADE ^etc/.*$ NO
UPGRADE ^etc/X11/.*$ YES
UPGRADE ^etc/X11/XF86Config$ NO
```

Заставят pkgadd никогда не модифицировать что-либо в /etc/ (включая подкаталоги), кроме файлов в /etc/X11/, если это не файл /etc/X11/XF86Config. При этом более поздняя запись имеет больший приоритет при запросе. Удалить пакет также просто: $ pkgrm bash – и его как не бывало. Узнать информацию об установленных пакаджах можно, набрав $ pkginfo -i, или, если нужен какой-то определенный, то $ pkginfo -l bash. Используя pkgmk, можно собрать пакаджи самому. И наконец, система портов, позволяющая собирать пакеты программы самому (при условии, что она установлена и находится в /opt). Правда, здесь понятие порта немного отличается от принятого во FreeBSD. Под портом понимается CVS-архив программ (ftp://cvsup.fukt.bth.se по умолчанию) и клиентская программа, способная загрузить их оттуда. Для начала обновляем дерево портов.

```
$ ports -u
Connected to cvsup.fukt.bth.se
Updating collection base/cvs
...
Updating collection opt/cvs
```

С помощью ports -d сверяем версии программ, установленных в системе и имеющихся в обновленной коллекции.

Например:

```
$ ports -d
No differences found
```

В этом случае можно ничего не делать – у вас самые свежие утилиты. Иначе, если найдено расхождение, то вводим:

```
$ cd /usr/ports/opt/xfree86
$ pkgmk -d
```

После чего загружается указанная программа с узла, указанного в Pkgfile, далее формируется пакет и теперь осталось, используя pkgadd, установить или обновить пакет. Хотя все это можно сделать за один шаг:

```
$ pkgmk -d -i
```

для установки или

```
$ pkgmk -d -u
```

для обновления имеющегося в системе.

Если процессор помощнее i686, то настоятельно советую заглянуть в файл /etc/pkgmk.conf; там найдете такие строки:

```
export CFLAGS="-O2 -march=i686 -pipe"
export CXXFLAGS="-O2 -march=i686 -pipe"
```

Замените значение переменных на требуемые, иначе все программы будут собираться под данный процессор, и о никакой оптимизации речи не будет.

Итак, найдет ли CRUX своего пользователя. Несомненно. Те, кто устал от всенастроености или хочет разобраться во всем с самого начала или просто иметь чистую систему без лишних невесть откуда взявшихся приложений, кто не хочет блуждать в дебрях каталогов, а хочет иметь простую и понятную файловую систему, кто сам привык разыскивать программы в Интернете, кто уже во всем для себя разобрался, кто использует десяток другой утилит, за развитием которых следит постоянно, кто хочет всегда иметь самую современную систему, оптимизированную под свое оборудование… Короче, все они найдут CRUX очень удобным инструментом. К тому же, исходя из опыта могу сказать, что Linux вообще-то устанавливается только один раз, остальное время он просто обновляется. В данном случае имеются все для этого. И еще один момент, о котором я как-то раньше и не думал – дистрибутив до того прост по своей структуре, что, используя его как основу, можно запросто создать свой локализованный CRUX. Жаль только, для этого нужен нормальный быстрый канал.

Достаток и уют

Сергей ЯРЕМЧУК, grinder@ua.fm

Ну наконец-то дождались. После чреды многочисленных предрелизов, которая, казалось, не закончится никогда, в конце января была таки выпущена следующая версия 3.1 самой популярной и любимой пользователями интегрированной рабочей среды KDE. Создатели дистрибутивов включили KDE в последние релизы своих ОС, пользователи всего мира устремились на сайт проекта http://www.kde.org за новой версией, и где-то там, среди тысяч и тысяч запросов пробивался, посылая IP-пакеты через перегруженный сервер, и я.

По большому счету, большинству пользователей достаточно установить программный минимум для нормальной работы системы и KDE со всеми сопутствующими приложениями, и этого будет вполне достаточно для повседневной деятельности. Данный проект уже настолько разросся, что в базовом комплекте имеются около 18 различных приложений, включая средства для разработки, утилиты администрирования системой и сетью, работы с мультимедиа, играми, Web-разработки и еще много чего. Плюс к этому – красивейший, удобный, легконастраиваемый и

привычный пользователю интерфейс, к тому же переведенный на 70 языков планеты (KDE 3.0.4 – 51 язык). В общем, понятно, откуда такая бешеная популярность. Проект KDE имеет 24 официальных зеркала в 16 странах, ftp-архив – 71 официальные зеркала в 30 странах.

И хотя работы над версией продолжаются, это уже полноценный, а не промежуточный (рабочий) релиз. По сравнению с предыдущей версией 3.0 было исправлено около 1000 ошибок и недочетов. CVS-архив KDE содержит приблизительно 2.6 миллионов строк кода (для сравнения, последнее ядро Linux серии 2.5 – около 3.1 миллиона). При разработке KDE одним из первостепенных требований было сделать самую безопасную настольную систему. Разработчики поощряют любого, кто найдет бреши в защите.

Итак, что же мы имеем на сегодняшний день?

Мой любимый Web-браузер **Konqueror** (http://www.konqueror.org), входящий в стандартную поставку, имеет компонентно-ориентированную архитектуру, которая объединяет функциональные возможности Internet Explorer/Netscape Communicator и Windows Explorer. Поддерживается полный комплект всех интернет-технологий, доступных на данный момент, включая JavaScript, Java, HTML 4.0 (почти полностью), CSS-1 и -2 (почти полностью, подробности на http://www.konqueror.org/css2), SSL (через OpenSSL), а также всевозможные плагины к Netscape Communicator, включая Flash, RealAudio, RealVideo. Если по какой-то причине сеанс оборвался, то Konqueror позволяет быстро восстановить его. Обновленный менеджер закачки **KGet** (http://kget.sourceforge.net), подобный GetRight и GO!Zilla, позволяет управлять любым количеством закачиваемых файлов, добавлять, удалять и приостанавливать загрузку, показывающий текущее состояние загрузки (прогресс, размер, скорость и расчетное время до завершения), интегрирован не только с Konqueror, но и с kdenetwork. Кроме того, следует вспомнить, что Konqueror – еще и прекраснейший файловый менеджер, который можно настроить для показа сразу нескольких каталогов, задать различные варианты отображения содержимого каталога, установить фон по вкусу (можно использовать уже заготовленные предустановки, а можно создавать свои). С помощью комбинации Ctrl+Shift+N теперь можно создать еще одно окно и получить таким образом возможность работы с несколькими документами в одном окне – так что Opera хвастаться уже, собственно, и нечем. К тому же Konqueror позволяет просматривать (только read-only) великое множество различных файлов, как графических, так и текстовых, созданных в различных пакетах, без вызова внешних приложений. Кстати, сам Konqueror непосредственно не поддерживает все эти форматы, это заслуга библиотек Kparts, позволяющих внедрять эту поддержку. Поэтому для просмотра графических файлов должен быть установлен **KView**, для PostScript – **Kghostview**, и, конечно же, **KOffice** берет на себя львиную долю форматов. Дополнительно теперь поддерживаются ряд плагинов для создания, редактирования и/или просмотра метаинформации различных типов файлов (например, аудиофайлов, файлов изображения, бинарных пакетов, данных записной книжки, электронной почты, файлов видео, конфигурационных файлов и языков программирования, MP3- и IDv3-тэги). Эта новая технология дополнена возможностью поиска в метаинформации файла с помощью поисковой утилиты диспетчера файлов, включая поиск в мультимедийных файлах. Теперь при включении режима предварительного просмотра файлов можно даже прочитать, что написано в маленьких текстовых файлах. Плюс ко всему полная настраиваемость интерфейса (прилагается конфигурационный файл в XML-формате), возможность убрать, например, лишние компоненты или эффекты, тем самым существенно облегчив его.

С помощью нового инструмента kio_fish обеспечивается возможность прозрачного безопасного доступа к сетевым ресурсам, включая Linux NFS, MS Windows SMB, HTTP-страницы, каталоги FTP и директории LDAP. Примененный промышленный стандарт SSH позволяет с надлежащим удостоверением подлинности, легко и безопасно манипулировать файлами, зафиксированными на удаленной машине, причем для этого на ней не нужно устанавливать дополнительное программное обеспечение. Вдобавок, модульный характер архитектуры KDE позволяет без особых сложностей подключать дополнительные протоколы (типа IPX или WebDAV), которые будут автоматически доступны всем приложениям KDE. Добавлены два вида

ненавязчивых всплывающих сообщения **KpassivePopup** в панели задач и **KWindowInfo** в области заголовка.

Второе место по значимости занимает KOffice, текущая версия 1.2.1. К нему я особой любви ранее не испытывал, но сейчас мнение полностью изменил. В состав KOffice входит восемь программ. В первую очередь это текстовый редактор Kword, умеющий многое из того, что должен уметь нормальный текстовый редактор и, главное, позволяющий нормально работать с не особо навороченными Word'овскими файлами. Также KSpread, неплохой табличный процессор для использования дома или в небольшом офисе, не требующий дальнейшего портирования документов в Excel (это касается и всего пакета в целом). Программа подготовки презентаций Kpresenter позволяет подготовить вполне приличную презентацию, есть и пока еще не очень развитый редактор векторной графики Karbon, приходящий на смену Kontour, а также две вспомогательные программы для создания определенных объектов в других приложениях: KChart для создания диаграмм и KFormula для работы с математическими формулами.

Почтовый клиент **Kmail** (http://kmail.kde.org) получил дополнительно поддержку важных протоколов защиты S/MIME, PGP/MIME и X.509v3. Кстати, проект защиты разрабатывается по контракту с немецким правительством. Плюс ко всему расширена поддержка OpenPGP (http://www.openpgp.org), и теперь Kmail автоматически может шифровать сообщения для каждого адресата отдельно.

Персональный органайзер и календарь **KOrganizer** (http://korganizer.kde.org), ныне оснащенный новым плагином для доступа к серверу Exchange 2000, поддерживает индустриальный стандарт iMIP, применяемый в Outlook, Lotus Notes и Evolution (включая функции передачи событий и todos), а также стандарт iCalendar протокола передачи задач. Ко всему прочему, улучшилась общая интеграция адресной книжки и добавились настраиваемые шаблоны для событий и todos, а также системы индикации аварии для todos.

KAddressbook получил дополнительно способность сбора информации о возможных контактах от одного или большего количества серверов LDAP; последние выводят сведения, обращаются к многочисленным записным книжкам и поддерживают импорт/экспорт промышленного стандарта vCards(включая и новый vCard 2.1). Кроме того, интерфейс пользователя принял ряд расширений применимости типа jump bar для быстрого упорядочивания по алфавиту доступа к участникам и к категориям участников. Поддерживаются плагины, позволяющие «на лету» добавлять новые функциональные возможности; всю информацию можно импортировать в базу данных собственного формата программы.

Расширилась поддержка Palm. В дополнение к функциям основного инструмента для работы с этими устройствами KPilot, некоторые приложения (например, KAdressbook) теперь способны импортировать информацию напрямую. Добавлена поддержка новых возможностей Time, AvantGo.

В этой версии появилась новая фича, позволяющая создать более управляемую среду – ее оценят владельцы интернет-кафе, системные администраторы и прочие заинтересованные в разделении прав пользователей лица. Она позволяет лишать указанного пользователя возможности осуществлять определенные действия (запускать некоторые приложения, менять обои, изменять программы в стартовом меню и пр.). Пока все это можно реализовать путем прямой модификации конфигурационных файлов, но в следующей версии должен появиться удобный графический инструмент.

Для тех, кто часто переключает рабочие места, KDE предлагает новый VNC-совместимый рабочий стол, разделяющий структуру (**KRfb**, **KRdc**). Пока что эта функция в основном предназначена для удаленной технической поддержки, чтобы удаленный пользователь мог видеть ваш рабочий стол и наблюдать за вашими действиями; но начиная с KDE 3.2 она позволит пользователям делить Рабочий стол между несколькими машинами в гетерогенных сетях.

Мультимедийный пакет **Kdemultimedia** получил новый видеодекодер, основанный на xine и позволяющий просматривать видео в форматах AVI, DivX, Cinepak, Sorenson Video, MPEG 1/2 and 4, QuickTime/MOV, ASF и многих других. Данный декодер встроен в файловый менеджер. Еще следует упомянуть KaudioCreator, новый инструмент, представляющий собой фронт-энд к

различным ripper'ам и аудиокодерам, позволяющий без труда сграбить понравившийся аудио-CD. Проигрыватель **Noatun** (http://noatun.kde.org) получил плагин, позволяющий разыскивать определенные мелодии.

Средство просмотра графических файлов KView теперь оснащено новым плагином для слайд-шоу, дополнительно улучшена поддержка колеса мыши и функция автоматического изменения размеров.

В обновленном KDE появился новый пакет Edutainment от **KDE Edutainment Project** (http://edu.kde.org), имеющий образовательную специфику. В настоящее время представлены соответствующие программы в четырех категориях: словари (FlashKard), языки, химия (Kalzium), математика (KmPlot, KPercentage). К сожалению, русский и украинский языки обойдены вниманием.

Чтобы скоротать время, можно обратиться к пакету **kdegames**, в котором добавлены новые игры: **Kolf** (http://www.katzbrown.com/kolf/) для любителей гольфа, **Atlantik** (http://www.unixcode.org/atlantik) — нечто наподобие «Монополии», с поддержкой сети и всевозможных оригинальных тем оформления, **blackjack** **Megami** (http://www.freekde.org/neil/megami) и аналогичная ей **Klickety** (http://klickety.sourceforge.net).

Для разработчиков Web-страниц припасен удобный **Quanta Plus** (http://quanta.sourceforge.net), поддерживающий HTML версии 4.0, CSS (Cascading Style Sheets), JavaScript (EMCAScript). Полная поддержка PHP, включая отладчик, подсветку синтаксиса, завершение названия, список переменных и функций. С помощью drag'n'drop можно запросто создать шаблон файла, рисунка или участка кода. Вдобавок, есть встроенный менеджер закачки на сервер. Очень удобная штука.

Многооконный редактор **Kate** (http://kate.kde.org) получил в придачу ко всем своим достоинствам улучшенную поддержку XML. Помимо уточнений к соответствующему плагину, в редактор добавлена функция проверки документов этого формата.

Терминальный эмулятор **Konsole** также подвергся значительным усовершенствованиям, позволяющим его оптимально настроить и комфортно работать с ним. Появилась возможность начинать сессию сразу с закладок URL (включая такие как ssh://user@host и telnet://host), улучшился механизм поиска в history (добавлена поддержка регулярных выражений), теперь можно с помощью пункта меню Copy копировать информацию во второй буфер обмена, также позволяется перенести любой ярлык приложения в окно терминала и вставить таким образом путь к нему со входом команд cp, ln и mv, каталог инициализации или текущий сохраняется теперь как часть сессии, появилась возможность отсоединить сеанс от главного окна.

Настроить внешний вид и поведение системы можно с помощью улучшенного, ставшего более удобным и понятным Центра Управления KDE (KControl), ныне более тесно интегрированного со справочной системой KhelpCenter. Настройке поддается практически все - наряду с использованием ставших уже привычными функций замены обоев, тем рабочего стола или звуковых схем событий, также можно выбрать группу иконок для приложений, изменить цвет заголовков окон. Панель теперь можно прикрепить к любому краю экрана, установить любой размер, как по горизонтали, так и по вертикали, выбрать рисунок для размещенных на ней кнопок или для всей панели в целом. В общем, рабочую обстановку можно полностью настроить по своему вкусу. Напомню и про удобное средство KAppfinder, позволяющее найти и добавить приложения в меню KDE. Переключаться между Рабочими столами теперь можно с помощью колеса прокрутки мыши. Могу сказать, что ребята из **KDE Usability Project** (http://usability.kde.org) свой хлеб ели не зря.

Нововведения можно перечислять и перечислять, настолько их много, но лучше один раз все-таки увидеть. Теперь буквально в двух словах о том, что нас ждет впереди (подробнее на http://developer.kde.org/development-versions/kde-3.2-features.html).

В следующем релизе 3.2 планируется добавить еще одно приложение в систему управления коллективной работой **Kontact** (http://www.kontact.org), интегрирующую в себе достижения двух проектов, работавших в этом направлении — **KDE PIM** (http://pim.kde.org) и **Kroupware** (http://www.kroupware.org). Ожидается появление такой фичи как mouse gestures, ныне доступной

пользователям Opera, позволяющей с помощью простых движений мыши выполнять определенные команды. В **Run Command** добавятся функции автодополнения, как в bash, и поддержка drag'n'drop. Также планируется постепенно отказаться от формата .wav и полностью перейти на Ogg Vorbis. В отдельных приложениях готовится поддержка LIRC (Linux Infrared Remote Control), позволяющая управлять ими с помощью пульта ДУ. Ksnapshot позволит делать снимки без декорации окна. Noatun получит эквалайзер в стиле K-Jofol.

Конечно, найдется пользователь, который будет утверждать, что Mozilla лучше Konqueror, что XMMS гибче и легче Noatun, что вместо Kmail удобнее использовать Sylpheed, а сам KDE гораздо прожорливее IceWM, но главный конек KDE – интеграция, и тут уже не поспоришь. Все приложения отлично взаимодействуют друг с другом, что дает дополнительные удобства для конечного пользователя. К тому же ему не придется собирать все необходимые для работы приложения по всему Интернету. Что тут говорить! Разработчики очередной раз подтвердили свой класс, а пользователи получили действительно удобную и приятную в работе среду.

Висит? Снимем!

Петр 'Roxton' СЕМИЛЕТОВ

Если зависает Windows, пользователь делает некие телодвижения, а затем, убедившись в «тщете и суетности этого мира», со спокойным сердце нажимает RESET. В Линуксе все обстоит иначе. Начнем с того, что «подвесить» Линукс вполне реально, несмотря на стабильность системы. «Подвесить» взято в кавычки потому, что при кажущихся признаках капитального останова системы ядро продолжает работать. Давайте разберем несколько типичных ситуаций и посмотрим, как можно с честью выходить из них. Предлагаю вам ощутить себя в нескольких героических ипостасях.

Ипостась первая — Зевс-Громовержец

Предположим, что вы работаете в KDE. У вас нет KDE? Ничего не могу поделать. KDE – самая популярная графическая оболочка, поэтому плясать мы будем вокруг нее. Итак, вы мирно работаете в KDE, запускаете какую-то шаманским образом откомпилированную прогу, и она повисает. Можно запустить менеджер процессов (Ctrl+Esc) и прибить замороженное приложение, но я предпочитаю более скоростной способ. Нажимаю Ctrl+Alt+Esc, и курсор превращается в молнию! Теперь достаточно щелкнуть ею по зависшему окну, и оно исчезнет.

Ипостась вторая – Киллер Икс

В него вам придется перевоплотиться, когда зависнут «иксы» - графическая оболочка. Или не зависнут, однако будут себя вести вызывающе и неадекватно. Например, самовольно уползет в свою нору мышь, не подавая больше признаков жизни, или оболочка упорно откажется закрыть себя и собственные окна. Знаете, ее можно понять – к десктопу приросла, — но ведь это противоестественно.

Рецепт прост, как ящик пива. В любой иксовой оболочке, будь то KDE, Gnome или BlackBox, срабатывает комбинация клавиш Ctrl+Alt+Backspace. Сразу после их нажатия следует низвержение экрана во тьму. Ничего страшного. Это всего лишь консоль. Иксы закрыты, можете снова их запускать – startx.

Ипостась третья - Терминатор

Могут произойти и совсем скверные вещи. Например, у вас компьютер подключен к бесперебойнику, а монитор – нет. Когда свет отрубают и наступает временный Апокалипсис, компьютер продолжает работать какое-то время, питаясь от аккумуляторов, а вот монитор гаснет. Между тем, вы же в поте лица своего трудились в KWord'e над неким громадным текстовым файлом и не успели его сохранить.

Положим, сохранить вы успеваете – вот оно, судорожно-резвое нажатие на Ctrl+S. Но ведь реальное сохранение могло не произойти! Линукс кэширует данные и периодически сбрасывает их на диск. Последняя операция носит титул синхронизации буфера. Ее можно запустить

вручную, нажав на Alt+SysRq+S. Напомню, что SysRq – это та же клавиша, что и PrintScreen. SysReq расшифровывается как System Request, то бишь «системный запрос». Итак, нажав вышеприведенную комбинацию, мы синхронизировали буфер.

Но ведь нужно еще выйти из системы как полагается. Ведь когда, допустим, вы выходите из дому, забыв выключить газ, свет и воду, то могут произойти всякие... Неприятные вещи, скажем так.

Хорошо, если в момент выключения монитора вы были в консоли, в ее командной строке - тогда вы можете быстро набрать вслепую команду halt, которая синхронизирует буфер, размонтирует файловые системы и, наконец, выключит питание компьютера. А если вы в графической оболочке? Если вы не помните ее «горячие клавиши» для выхода или перезагрузки (если оболочка вообще поддерживает таковые)?

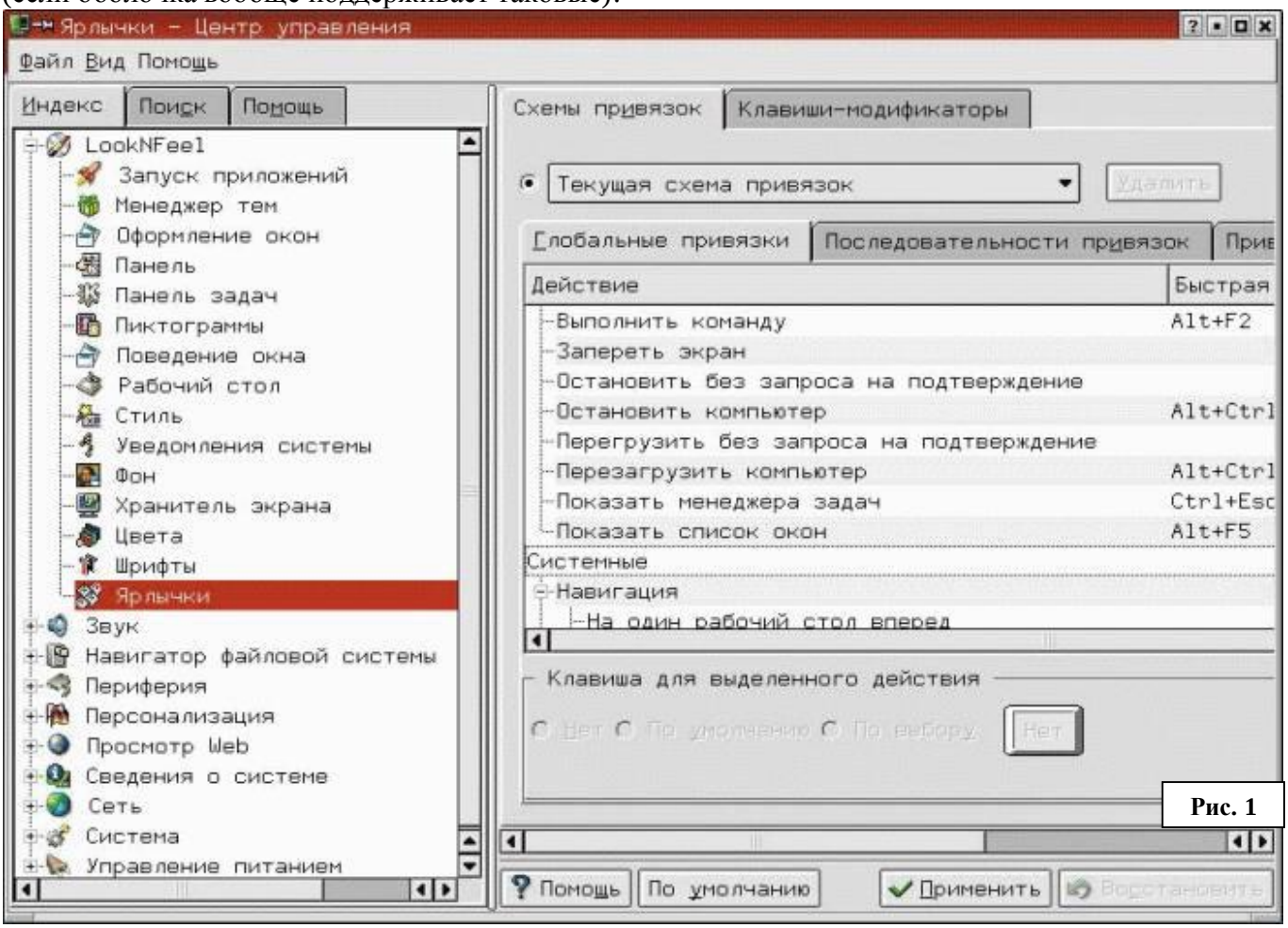


Рис. 1

Здесь мы снова сделаем уклон в сторону KDE. Есть, есть в ней такие волшебные клавиши. Нужно только их настроить. Для этого запустите Центр управления KDE и перейдите в нем в LookNFeel > Ярлычки. Теперь, на странице Глобальные привязки найдите пункт Остановить без запроса на подтверждение и назначьте ему свое сочетание клавиш (Рис. 1).

Но это в KDE. А нет ли чего-нибудь глобального, для других файловых менеджеров. И чтобы в консоли это тоже работало? Есть! Можете записать себе на бумажке или вырезать кусочек статьи и присобачить его на стенку. А еще лучше – держите весь журнал под рукой. Итак, список волшебных клавиш, действующих во время шторма:

- **Alt+SysRq+O** – действует на систему подобно выстрелу в висок. Никакой синхронизации буфера, размонтирования и прочего. Сразу гасится свет, то бишь вырубается питание. Поэтому до нажатия этих клавиш следует синхронизировать буфер (Alt+SysRq+S), затем размонтировать файловые системы (Alt+SysRq+U) и только потом нажимать страшное Alt+SysRq+O. И запомните – сначала синхронизация, потом размонтирование, однако не наоборот;

- **Alt+SysRq+K** – эта комбинация носит название Secure Access Key (SAK). Она убивает все запущенные в текущей виртуальной консоли процессы. Даже глазом моргнуть не успеете. Полезно использовать, когда зависла некая программа и даже не позволяет вам переключиться в другую виртуальную консоль. Исторически SAK предназначался для других целей. Применять его советовали пользователям перед тем как залогиниться в систему. Дело в том, что довольно легко написать программу или скрипт, эмулирующие логин. Неискушенный пользователь вводит свой логин и пароль, а «троянский конь» сохраняет их в надежном месте, чтобы злоумышленник воспользовался плодами вашей доверчивости. Если же юзер перед тем как зайти в систему нажмет SAK, то прибьет трояна и получит нормальное, «оригинальное» приглашение системы.

А вот еще три сходные по действию функции-киллеры:

- **Alt+SysRq+E** – посылает сигнал SIGTERM всем процессам, кроме init. Сигнал SIGTERM означает, что система завершает некий процесс и прибирает за ним мусор – закрывает открытые процессом файлы, убивает временные и так далее;
- **Alt+SysRq+I** – посылает сигнал SIGKILL всем процессам, кроме init. В отличие от предыдущего сигнала, SIGKILL грубо прерывает выполнение программы, не выполняя после этого «зачистку».

Здесь уместно сделать небольшое отступление и рассказать немного о сигналах, с помощью которых можно управлять выполнением программ, а точнее – их завершением. Сначала нам нужно получить PID программы. Аббревиатуру PID следует толковать как Process Identifier, то бишь идентификатор процесса. Наберем в консоли следующую команду:

```
ps
```

После ее выполнения на экран будет выведена информация о запущенных в текущей консоли процессах. Примерно вот так:

```
[root@localhost root]# ps
PID TTY      TIME CMD
1434 pts/2  00:00:00 bash
1621 pts/2  00:00:00 bc
1622 pts/2  00:00:00 ps
```

Из этого следует, что в во время запуска команды ps были активны следующие процессы: bash(интерпретатор оболочки), сама ps и удобный консольный калькулятор bc. В первом столбце таблицы видим интересующие нас идентификаторы процессов. Зная их, можем послать процессу некий сигнал. Допустим, мы хотим прервать выполнение bc. Попробуем сделать это вежливо, послав SIGTERM:

```
kill -SIGTERM 1621
```

В ответ — молчание... Проверяем активные процессы командой ps.

```
[root@localhost root]# ps
PID TTY      TIME CMD
1434 pts/2  00:00:00 bash
1621 pts/2  00:00:00 bc
1647 pts/2  00:00:00 ps
```

Как видим, калькулятор все еще висит в памяти. Тогда попросим его более настойчиво, уже с помощью сигнала SIGKILL:

```
kill -SIGKILL 1621
```

И что же? На экране появляется сообщение:

```
[1]+ Killed bc
```

С чем и поздравляю – процесс убит наповал. Вот что называется «магией на кончиках пальцев». Далее:

- **Alt+SysRq+L** – посылает сигнал SIGKILL всем процессам, включая init. В этом случае система уходит в полный даун. Эту команду лучше не трогать. Смерть от напалма!
- **Alt+SysRq+B** – перезагрузка системы. Очень быстрая, потому что этапы синхронизации и размонтирования пропускаются. Употреблять только в самых крайних случаях. С таким же успехом можете нажать на Reset.

- **Alt+SysRq+R** – изменение режима клавиатуры с Raw в XLAT. Например, после краха видеоплейера или игры, работающего с популярной библиотекой SVGALib, вы могли заметить, что консоль не реагирует на нажатия клавиш. Вот это средство и поможет!

Под конец статьи затрону тему, которая напрямую связана с проблемами функционирования программ. Говоря проще – с вылетами. Речь идет о coredump'ах – файлах довольно ощутимых размеров, которые создаются после аварийного завершения программы. Читать coredump'ы может быть интересно программистам, для простых же пользователей они совершенно бесполезны и только засоряют винчестер. Как бороться?

Открываем в текстовом редакторе файл bashrc, который лежит в директории /etc. В его конец добавляем всего одну строчку:

```
ulimit -c 0
```

Этим мы ограничиваем размер создаваемых coredump'ов до нуля, тем самым предотвращая их появление. Чтобы команда вступила в силу, надо (в общем случае) перезагрузить систему. Проверим, получилось ли. Пример даю исходя из того, что вы работаете в KDE. Заходим в директорию с каким-нибудь текстовым файлом, из контекстного меню выбираем просмотр этого файла неким редактором. Теперь нажимаем **Ctrl+Esc**, чтобы вызывать **Системный монитор** (Рис. 2). В его списке делаем правый клик на имени редактора, чтобы вызвать контекстное меню. В нем активизируем пункт **Послать сигнал**. Появится меню сигналов, из которых следует выбрать SIGSEVG. Программа «вылетает», а мы смотрим, появился ли coredump. Не появился. Очень хорошо.

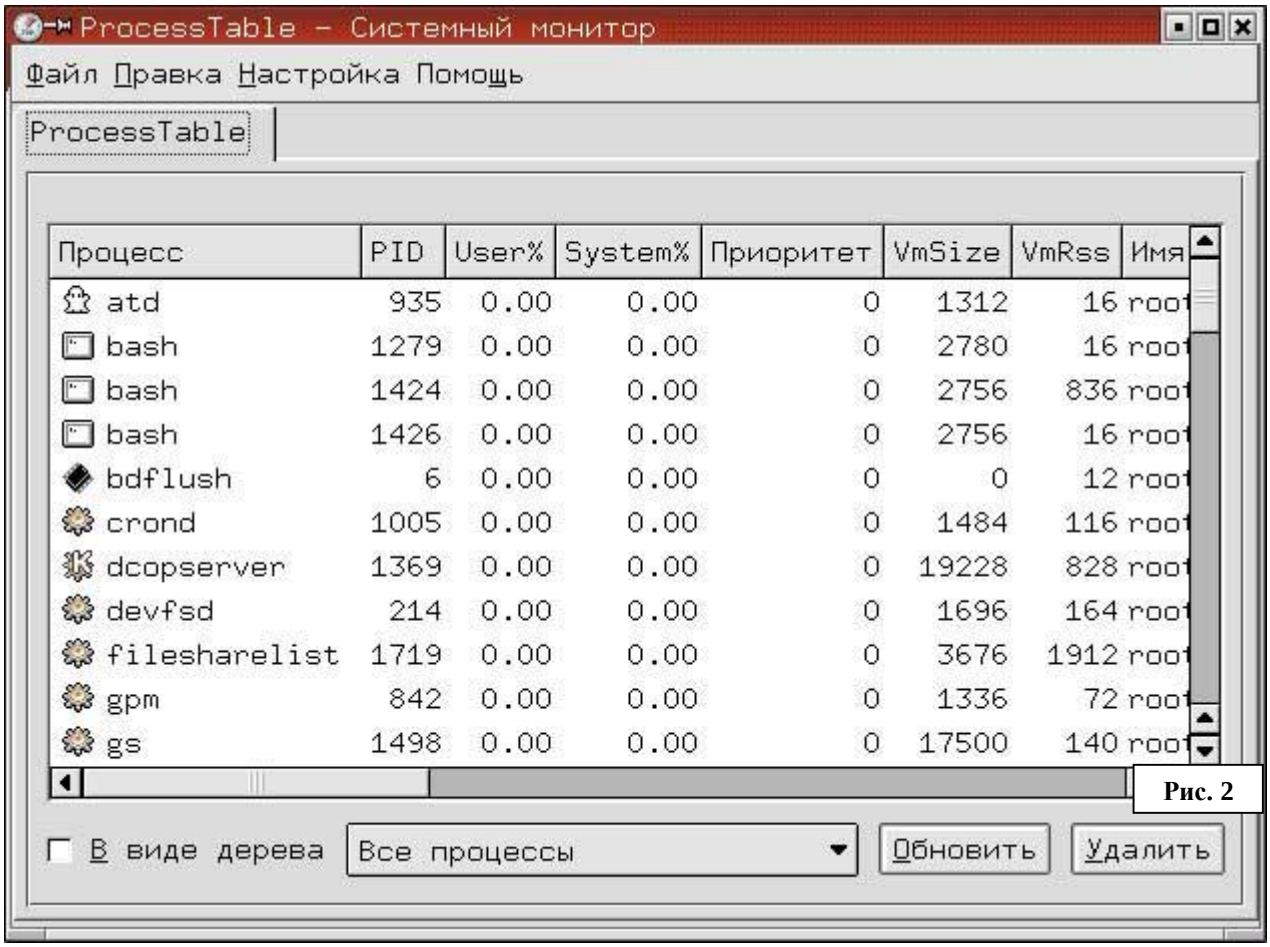


Рис. 2

Как поют пингвины

Сергей ЯРЕМЧУК, grinder@ua.fm

Уже больше года я рассказываю о трудностях и лишениях командной строки, показываю как привлекательные, так и спорные стороны Linux. За это время на мой адрес пришло много писем. Если поначалу были и такие, в которых писалось что-то вроде «все понятно, но я не понял, что такое Linux. Это программа такая, да?», то теперь когда мы в основном уже разобрались с повседневной деятельностью, читателя интересуют вопросы уже более специфического использования этой системы. Например, можно ли создавать музыку под этой операционной системой?

Да, задачка еще та. Я когда-то давно, до моего знакомства с этой осью, юзал SoundForge и CoolEdit, но потихоньку опингвинившись, как-то забросил это дело и увлекся сетями. К тому же в дистрибутивах не было (нет и сейчас) и намека на программы для обработки звука. Да что там говорить – звуковую карту настроить было проблематично!

Но прогресс на месте не стоит, и Linux потихоньку двигается к пользователю. И вот когда мне первый раз задали такой вопрос, честно говоря, застал он меня врасплох. Я начал потихоньку копать в этом направлении. Копал довольно долго (где-то с сентября 2002). Вырисовалась довольно интересная ситуация, которую я сейчас вам и обрисую. Получается, что общей информации по обработке звука кот наплакал, но если начинать раскапывать отдельные приложения, то ее получается даже слишком много. Беда только, она беспорядочна – поэтому, наверное, я и не затрагивал так долго эту тему. Linux первоначально задумывался всего лишь как операционная система, имитирующая Unix, а кто, скажите мне под Unix'ами, музыку создает? Что-то леса рук не видно. Вот именно отсюда и проблемы. Система понравилась программистам-сисадминам, что и сказалось на предлагаемом программном обеспечении. Дошло до того, что количество различных простых редакторов а-ля vi или nano уже давно перевалило за тысячу. Ведь кто пишет программы в этой системе? Сами же пользователи, если понадобилось что-то срочно. Например, захотелось редактор с горячими клавишами, имитирующими любимый MSWord :-) – бах... и появился еще один редактор. Это как раз тот случай, когда свобода пошла не совсем на пользу, налицо явный перекос в приложениях. То есть для того чтобы появилось такое приложение необходимо, чтобы музыкант поставил себе Linux, полюбил его, как я, выучил C и наваял себе пару програмулин. Вы себе можете это представить? Нет. Вот и я с трудом. Таких людей единицы, поэтому прогресс в этом направлении хоть и двигался (прошу заметить, глагол употреблен в прошедшем времени), но очень уж медленно. Но это еще не самое страшное. Первое, что стояло на пути прогресса в этом направлении, – это сама система. Главное для сервера что? Стабильность. Которая никак не вязалась с необходимостью обрабатывать звук в реальном времени. О проблемах ядра и путях их решения я писал в отдельной статье, еще некоторые подробности можно найти по адресу http://www.linuxdj.com/audio/lad/resourceslatency.php3. Так что этот вопрос будем считать уже решенным. При необходимости, использовав один из патчей, можно выжать из ядра честные 2 мс, чего вполне достаточно. А написав приложение под real-time ядро или хотя бы под те, что используют FIFO_SHED, можно получить и меньшие задержки. И кстати, по официальной информации, компания ALTLinux до недавнего времени собирала ядра именно с таким патчем, это сейчас они там что-то такое замышляют и собирают пока ядра без него.

Другая проблема состоит в том, что в настоящее время используется несколько драйверов звуковых карт. По умолчанию Linux работает со свободными аудиодрайверами OSS/Lite aka OSS/Free aka OSS/Kernel от **Open Sound Sistem** (http://www.opensound.com/), есть и коммерческие версии с большей поддержкой разнообразного оборудования и дополнительными возможностями. Это первые драйвера, которые вообще появились в этой системе, бесплатная версия их включена в ядро и, наверное, наиболее популярна. Но вот latency их – около 150 мс, что, мягко говоря, многовато будет. Другая альтернатива, проект **ALSA** (Advanced Linux Sound Architecture, http://www.alsa-project.org), выросший из Gravis UltraSound Project. Кто, может, помнит, были когда то такие звуковухи, вот первоначально под них и писались драйвера. Со временем проект расширился, и в настоящее время он предусматривает поддержку большинства карт, а главное,

полную совместимость с OSS/Lite (хотя, если честно, глюки бывают (или бывали)). Главное преимущество ALSA – поддержка одновременной работы нескольких звуковых карт, поддержка работы сразу нескольких приложений и возможность real-time режима. Практически во все современные дистрибутивы уже включены ALSA-драйверы. И latency уже получше будет – по утверждению разработчиков, не больше 5-6 мс, что является необходимым минимумом. И главное, что в следующей версии ядер 2.6.* ALSA-драйвера будут встроены в ядро, так что об OSS/Lite в скором времени, скорее всего, все потихоньку позабудут. Есть еще закрытые драйвера от производителей звуковых карт. Яркий пример – Creative Live, которые распространяются исключительно в бинарниках. Ничего не поделаешь, что есть, то есть.

Думаете, это все? А вот и нет. Есть еще такая штука – аудиосервер, позволяющий предоставить доступ к одному устройству /dev/dsp сразу нескольким приложениям. Напомню, OSS разрабатывалась лишь затем, чтобы обеспечить доступ к устройству, не более того, а потому одно приложение захватит его и не будет делиться остальными. За все время существования этой ОС этих серверов наплодилось такое количество! Каждый уважающий себя оконный менеджер обязательно имеет свой звуковой сервер: в KDE рулит aRts, в Gnome – **ESD** (esound или Enlightenment Sound Daemon), в WindowMaker – WMSound, есть свой звуковой сервер даже в малыше IceWM. Ситуация запутанная. К тому же сейчас вместо ESD, имеющего довольно скудные возможности, потихоньку продвигается «advanced» sound daemon (http://asd.sourceforge.net). Но мало того, что их наплодилось так много - все они задумывались для чего угодно, кроме как для обработки звука, а посему архитектура их далека от идеала и, как следствие, велики задержки. Обращает внимание разве что aRts; ребята из KDE в третьей версии немного подрихтовали его и теперь утверждают, что не такие уж большие у него эти самые задержки – 1 мс. Не знаю, верить или нет, мне он все равно кажется жутким тормозом, особенно по части потребления системных ресурсов. К тому же слишком много лишнего, как для обработки звука. Загрузите для интереса XMMS и NoAtun, а потом посмотрите при помощи top, как они потребляют ресурсы - второй в два раза тяжелее именно из-за aRts.

Но не все так мрачно. Наверное, решив, что чем что-то переписывать, лучше сделать с нуля, ребята из **LAD** (Linux Audio Development, http://www.linuxdj.com) создали аудиосервер **JACK** (Jack Audio Connection Kit, http://jackit.sourceforge.net), направив все усилия на то, чтобы обеспечить синхронное выполнение задач всеми клиентами и низкую латентность. Его можно использовать как с приложениями, написанными под low-latency, так и отдельно. При этом JACK вовсю использует SCHED_FIFO, и для стопроцентного real-time желательно запускать от имени root'a (или же открыть допуск к CAP_RESOURCE), что не всегда безопасно. Можно, конечно, использовать suid, но при этом может понадобиться изменить пару строк в исходниках ядра. А так сервер получился действительно шустрый и ненакладный для системы. Разработчики ясно дают понять, что это не замена для esd или artsd, они, мол, не старались предоставить многопользовательский доступ к устройству, их усилия направлены на создание многоприкладного устройства и разделение данных. И хотя работать с их изделием непросто, без него не откомпилируется ни одна уважающая себя программа, предназначенная для обработки звука, но к сожалению, пока таких единицы. И кстати, JACK работает только с ALSA, а по сути является его собственным звуковым сервером.

И еще одна проблема – мало просто создать приложение, необходимо обеспечить его максимальную функциональность т.е. нашпиговать его различными синтезаторами, эффектами, фильтрами, эквалайзерами и прочей дребеденью. Малыми силами это чересчур хлопотно, поэтому чаще всего эти модули отдают на откуп отдельным разработчикам и подключают потом в виде плагинов. Яркий пример – GIMP и XMMS; без подключаемых модулей это были довольно унылые программулины. Под Windows такого добра пруд пруди (хоть и стоят некоторые хорошую копейку), а вот в Linux маловато будет. Да, что-то из VST-плагинов втихаря уже портировано, но такого разнообразия, увы, пока нет. Чтобы решить проблему, было начато несколько проектов, но пока серьезно можно рассматривать разве только **LADSPA** (Linux Audio Developer's Simple Plugin API, http://www.ladspa.org) зародившийся в 2000 году (как и многие другие подобные проекты – видать, переломным был год); практически все новые разработки уже поддерживают эту

технологию. На ранней стадии проекта LADSPA неутомимые ребята из той же группы LAD вошли в контакт с компанией Steinberg для переговоров относительно переноса VST-архитектуры в Linux; увы, они получили отворот-поворот (правда, там кроме лицензионных проблем была еще проблема портировки интерфейсов, используемых в API Steinberg). А так, прошу заметить, эта архитектура полностью открыта и бесплатна, развивается довольно быстро, сейчас число плагинов в LADSPA уже перевалило за сотню, и в скором времени уже просто не будет необходимости в портировании VST. Подробнее о настройках и возможностях LADSPA можно почитать в статье Dave Phillips'a Linux Audio Plug-Ins: A Look Into LADSPA (www.oreillynet.com/pub/a/linux/2001/02/02/ladspa.html), на этом же сайте можно найти еще несколько полезных статей на эту тему. И кстати, имеется ladspa-плагин к XMMS, который можно найти по адресу www.ecs.soton.ac.uk/~njl98r/code/ladspa. Еще за плагинами можно заглянуть по адресу http://plugin.org.uk.

Следующая проблема, с которой сталкиваются абсолютно все, — установка программ. Подчас приходится решать довольно нетривиальные задачи. Мало того, что придется весь Интернет излазить в поисках как самих программ, так и недостающих библиотек, но потом все это нужно умело откомпилировать, наложить патчи на ядро и пересобрать его, не говоря уже о постоянном чтении man'ов для постинсталляционной настройки. При некоторые проекты ведут себя так, будто их программа распространяется исключительно в исходниках. И они по-своему правы, ведь только в этом случае достигается максимальная производительность, но не каждому пользователю это под силу, особенно если он только начал знакомиться с этой системой.

Естественно, никто не захочет с этим всем возится и будет пользоваться Windows или Mac'ом. Хотелось, чтобы было как там – вставил диск, откинулся на спинку кресла, в результате через некоторое время получил полностью настроенную систему со всеми необходимыми компонентами, чтобы после перезагрузки мог уже творить. Мне известны по крайней мере три проекта, нацеленных конкретно на то, чтобы обеспечить пользователя полноценной звукообрабатывающей системой, не особо при этом его напрягая. Два проекта находятся еще в стадии развития, а один, скорее всего, немножечко подзахлох. Под номером первым в этом списке – хотя бы потому, что после его объявления на Linux стали смотреть немножко иначе, — я бы поставил проект **AGNULA** (A GNU/Linux Audio distribution, http://www.agnula.org), стартовавший первого апреля 2002 года (это не шутка) и рассчитанный на два года. Как следует из названия, цель разработчиков – создать дистрибутив, ориентированный конкретно на музыкантов (и наполненный, кроме всего прочего, еще и соответствующей документацией). Сам проект проводится под эгидой Европейской комиссии, при участии множества крупных научных, общественных и коммерческих организаций, включая даже такие организации как французский институт акустических исследований **IRCAM** (http://www.ircam.fr), Music Technology Group (http://www.iua.upf.es/mtg), Music Acoustics Group (http://www.speech.kth.se/music/performance), Фонд свободного ПО (Free Software Foundation Europe, http://fsfeurope.org), даже французское отделение компании Red Hat тоже засветилось здесь. Как видите, люди собрались серьезные, серьезным обещает быть и результат. В результате планируется создать два варианта дистрибутива: Debian-вариант (**DeMuDi** — Debian Music Distribution, http://www.demudi.org), ориентированный скорее на специалистов, и Red-Hat версия (**ReHMuDi** — Red Hat Music Distribution) – для обычных пользователей. Различия между ними будут заключаться только в дистрибутивах-носителях, а также в особенностях установки и работы с ними, музыкальные же функции будут идентичны. На втором году работы, кроме того, планируется перенести проект также на PowerPC и 64-битную архитектуру. По секрету скажу, что основой ReHMuDi служит RedHat 7.3, у которого убраны все серверы (зачем музыканту Apache?) и прочая ненужная дребедень, а добавлены необходимые пакеты. Поэтому тем, у кого аналогичный дистрибутив (или совместимый), можно, не стесняясь, тягать пакеты по одному и устанавливать, а если будут жалобы на отсутствие чего-то, это что-то брать здесь же. Все нужные пакеты для удобства сложены в одну папку и доступны как в сырцах (scr.rpm), так и уже откомпиленные – i386.rpm. Правда, некоторые версии далеко не последние, но потом поверх можно будет преспокойно установить свежатину. Поверьте, мороки все равно будет меньше.

Следующий проект, о котором мне известно, – это ALT Linux Multimedia, который задумывается как дополнение к Junior и будет включать отдельный диск, на котором найдете:

- локализованное ПО для работы с мультимедийными данными;
- документацию;
- свободный клипарт (3D, SF2, ноты).

Больше и сказать нечего. Очевидно, есть смысл при выборе дистрибутива остановиться на ALT Linux, к тому же в их Сизифе уже имеются некоторые программы для обработки звука, как правило, локализованные и устанавливающиеся с одного тычка. Да, еще один момент. В настоящее время ведутся работы по украинизации многих приложений, в том числе и описываемых. Причем, как вы понимаете, когда есть исходный текст, проблема состоит именно в терминах, а не в необходимости декомпилировать код программы. Если кто может реально помочь в переводе специальных терминов на рідну мову, загляніть будь ласка за адресою http://dict.linux.org.ua или в рассылку linux@linux.org.ua, где обитают переводчики. Может, впишете свое имя в историю.

И последний проект, предлагающий комплексное решение означенной проблемы, — Turn-Key Linux Audio от Eastman Computer Music Center (http://lulu.esm.rochester.edu/kevine/turnkey), который первоначально задумывался как способ обеспечить бедных западных студентов ECMC полноценным студийным арсеналом, для использования на их домашних системах – как вы понимаете, с такой благородной задачей может справиться только linux и программное обеспечение с открытым кодом.

Здесь все просто (во, сказал!) – авторы собрали со всего света любимые утилиты для обработки звука, библиотеки, добавили кое-какую документацию и обеспечили удобство установки – достаточно запустить всего один скрипт, так же можно устанавливать и отдельные приложения. И после того, как все это хозяйство откомпилируется, пользователь, кроме документов и необходимых для работы библиотек, в итоге получает: 6 звуковых редакторов, 6 синтезаторов, около 5 миксеров, 7 анализаторов и еще большую кучу утилит, позволяющих содрать треки с Audio CD, записать их обратно, тьму конвертеров в различные звуковые форматы, утилиты поиска и море различных шаблонов, макрокоманд и программ, всех не перечислить. Единственный недостаток – размер, 116 Мб. Правда, и некоторые утилиты уже не последнего релиза, но при необходимости они довольно легко обновляются. Кстати – поверьте, это правда, – некоторые я смог увидеть работающими только здесь. Есть облегченная версия TKLA, но по сравнению с полной она выглядит чуть не издевательством. Авторы утверждают, что тестировали под Mandrake 8.x и 9.0, Suse, я устанавливал под RedHat 7.3 и потом под CRUX – идет на ура.

Среди коллекций программного обеспечения стоит также отметить PlanetCCRMA (http://ccrma-www.stanford.edu/planetccrma/software), научный центр по компьютерному исследованию в области звука и музыки Стэнфордского университета (CCRMA, Stanford University's Center for Computer Research in Music and Acoustics). У них довольно приличная коллекция собранных в rpm-пакеты low-latency ядер, свежих ALSA-драйверов и библиотек, а также различного музыкального, миди-, аудио-, видео- и 3D-софта с подробными мануалами и краткими инструкциями по установке. Так что если с make не в ладах, вам только сюда. Это, кстати яркий пример того, как можно заработать на открытом ПО, – они обучают пользованию за деньги.

Еще одна примечательная страница от Dave Phillips http://www.bright.net/~dlphilp/linuxsound – этот человек начал собирать здесь различные программы для обработки звука еще с 1995 года. Зайдите, не пожалеете – все ссылки рассортированы, собраны в одном месте, не нужно бегать по всему Интернету в поиске нужного приложения. Прекомпилированные RPM low-latency ядра, но с патчем от Ingo (не знаю, как по батюшке), можно найти на сайте Udo Jocher'a по адресу www.gardena.net/benno/linux/ (сами патчи — http://people.redhat.com/mingo/lowlatency-patches). Правда, первые версии патча сильно поругивали за низкую стабильность и надежность — «уродливые, но эффективные», даже дядюшка Линус покричал, – но сейчас, вроде, все путем. По адресу http://www.djcj.org/LAU/guide/ найдете Linux Audio Users Guide, в котором собраны ссылки на документы в виде HOWTO, которые могут понадобиться при настройке своей music-

workstation. И еще один интересный документ – The Mandrake Audio Workstation HowTo (http://www.desktoplinux.com/articles/AT8018846552.html), в котором рассказано, как превратить свой Mandrake в звукообрабатывающую студию в кратчайшие сроки, объяснено, почему Linux, почему Mandrake, даны рекомендации по оборудованию, приложениям и файловой системе (по мнению автора, оптимальна XFS – думаю, он прав, хотя RaiserFS тоже рулез).

Вот такие вот пироги с котятами. Если бы у меня этак годик назад спросили, что я думаю по этому вопросу, послал бы, наверное, куда подальше и попросил более не отрывать от серьезных раздумий. Но сейчас я уверен, что в ближайший год, максимум два, кое-кому придется немного потесниться. Не знаю, как насчет профессиональной студии, но качественную демку уже сейчас можно наваять под Linux без особых проблем. На сегодня все. Но тема не забыта, об отдельных утилитах будем говорить в последующих статьях, которые начну писать, как только поборю природную лень. Лето, однако.

Р.Маланчук, 2005 (ред.)